KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Exploratory Visualization Design towards Online Social Network Privacy and Data Literacy

Bo Gao

Dissertation presented in partial
fulfillment of the requirements for the
degree of Doctor in Engineering Science:
Computer Science

November 2015

Exploratory Visualization Design towards Online Social Network Privacy and Data Literacy

Bo GAO

Examination committee:
Prof. dr. Willy Sansen, chair
Prof. dr. Bettina Berendt, supervisor
Prof. dr. Philip Dutré
Prof. dr. Siegfried Nijssen
Prof. dr. Joaquin Vanschoren
Prof. dr. Bart Goethals
(Universiteit Antwerpen)
Prof. dr. Jo Pierson
(Vrije Universiteit Brussel)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science: Computer Science

November 2015

Abstract

People are documenting themselves and/or are being documented digitally. Large amounts of data are being generated and aggregated. This creates both challenges and opportunities. The sheer amount, complexity and (intentional) obscurity of data hinder people from understanding it. This in turn creates obstacles for people to become independent individuals and competent citizens. We, as researchers and practitioners, aspire to make tools that help people explore and understand the data collected about themselves, and help them make more informed decisions.

In Online Social Networks (OSNs), because of the difficulty in managing many online friends, users often experience the privacy problem "context collision", where their posts are seen by undesired audiences. This may not only lead to awkwardness or embarrassment, users are also disempowered in their online identity construction. We addressed this problem by investigating "friend-grouping" strategies, comparing community detection algorithms and developing an exploratory ego-network visualization tool, named FreeBu.

Also, as required by privacy-as-practice and critical data literacy, it is equally important to help a user gain insight into the behavior of user groups that are beyond ego-networks. More specifically, we investigated how Facebook users with different characteristics express sentiments, and how the texts posted on Facebook with different privacy settings exhibit different sentiments. On the one hand, we drew the links between our hypothesis testing and existing sociological research. On the other hand, we proposed and implemented algorithms that extract subgroup comparisons for the purpose of exploratory data analysis.

Furthermore, as individuals' activities are recorded and collected by companies, governments or other organizations, discriminatory decisions could be made against them. We looked into the field of Discrimination-aware Data Mining (DaDM). More specifically, we leveraged existing DaDM techniques, proposed meta-level measures and developed an exploratory visualization tool, named

D-Explorer, that can help people explore and better understand discriminatory patterns.

Finally, we compared and summarized state-of-the-art Javascript data-visualization libraries in terms of application domains, abstraction levels, visualization tasks and design patterns. We further translated visualization taxonomies to library design requirements, in order to comprehensively capture different aspects of data-visualization development. We also proposed improvements for future data-visualization library design.

In sum, this thesis took an interdisciplinary approach towards OSN privacy and aggregate data literacy. We iteratively designed and implemented two exploratory visualization tools. During this process, we leveraged the knowledge from the fields of information visualization, software engineering, human computer interaction, data mining and microsociology.

Beknopte samenvatting

Mensen documenteren zichzelf en/of worden digitaal gedocumenteerd. Grote hoeveelheden gegevens worden hierbij gegenereerd en samengevoegd. Dit creëert zowel uitdagingen en kansen. De enorme hoeveelheid, complexiteit en (opzettelijke) onduidelijkheid van deze gegevens belemmeren mensen om ze ten volle te begrijpen. Dit op zijn beurt leidt tot belemmeringen voor mensen om onafhankelijke individuen en competente burgers te zijn. Wij, als onderzoekers en praktijkmensen, streven naar hulpmiddelen die mensen helpen om de verzamelde gegeven over zichzelf te ontdekken en begrijpen, en hen te helpen beter geïnformeerde beslissingen te maken.

In Online Sociale Netwerken (OSNs), vanwege de moeilijkheid in het beheer van grote aantallen online vrienden, hebben gebruikers vaak last van het privacy probleem "contextuele botsing", waar hun berichten worden gezien door een ongewenste publiek. Dit kan niet alleen leiden tot genante of schaamtelijke situaties, het ontkracht gebruikers ook in de constructie van hun online identiteit. We pakken dit probleem aan door "vriend-groepering" strategieën te onderzoeken, doormiddel van het vergelijken van community detection algoritmen en het ontwikkelen van een ego-netwerk visualisatie tool, genaamd FreeBu.

Ook, zoals vereist door privacy-in-praktijk en kritische data literatuur, is het belangrijk om een gebruiker te helpen inzicht te krijgen in andere groepen dan het ego-netwerk. Meer in het bijzonder onderzochten we hoe Facebook-gebruikers met verschillende eigenschappen hun gevoelens uiten, en hoe de teksten gepost op Facebook met verschillende privacy-instellingen verschillende gevoelens vertonen. Aan de ene kant trokken we verbanden tussen onze geteste hypotheses en bestaand sociologisch onderzoek. Aan de andere kant hebben we algoritmen ontwikkeld die subgroep vergelijkingen extraheren met als doel het exploratief verkennen van data.

Bovendien, aangezien de activiteiten van individuen worden geregistreerd

door bedrijven, overheden en andere organisaties, kunnen discriminerende beslissingen gemaakt worden tegen hen. We keken naar het onderzoeksgebied van discriminatie-bewuste Data Mining (DaDM). Meer specifiek, we gebruikten bestaande DaDM technieken, stelden meta-level maten voor en ontwikkelden een verkennend visualisatie tool, genaamd D-Explorer, die mensen kunnen helpen om discriminerende patronen te ontdekken en beter te begrijpen.

Tot slot vergeleken we en maakten we een samenvatting van de state-of-the-art Javascript data-visualisatie bibliotheken in termen van toepassingsdomeinen, abstractie niveaus, visualisatie taken en design patterns. We vertaalden ook visualisatie taxonomieën naar bibliotheek ontwerp eisen, om de verschillende aspecten van data-visualisatie ontwikkeling volledig vast te leggen. We stelden ook verbeteringen voor, voor het ontwerp van toekomstige data-visualisatie bibliotheken.

Kortom, dit proefschrift heeft een interdisciplinaire benadering gehanteerd van OSN privacy en het begrip van geaggregeerde gegevens. We ontworpen en implementeerden twee verkennende visualisatiehulpmiddelen in een iteratieve manier. Tijdens dit proces hebben kennis van de vakgebieden van informatie visualisatie, software engineering, mens-computer interactie, data mining en microsociologie gebruikt.

Abbreviations

DaDM Discrimination-aware Data Mining
DD Direct Discrimination
D-Explorer Discriminatory ruleset Explorer
FA Feedback and Awareness
FreeBu Friend tree Bubbles
FSL Facebook Smart Lists
GMF Generative Model for Friendships
HMOD Hierarchical MODularity-based community detection
IDD Indirect Discrimination
IL Information Literacy
IV/Infovis Information Visualization
MOD MODularity-based community detection
MMOD Multi-membership MODularity-based community detection
OSN Online Social Network
OSNs Online Social Networks
PD Potentially Discriminatory
PND Potentially Non-Discriminatory
PFA Privacy-relevant Feedback and Awareness
SV/Scivis Scientific Visualization
Vis Visualization

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Every two days, we create as much data as we did from the beginning of time until 2003". "In five years, the amount of digital information will be at least ten times of what we have now".[1] Though having grown accustomed to facts or claims like these, as citizens and individuals, we seem lost in the sea of data. Meanwhile, various kinds of conveniently available information are competing for our attention. We are living in an age where data is abundant, information is widely accessible, yet knowledge is still difficult to acquire, as ever.

Within a short period of time, online services like Google, Facebook, Twitter, Weibo, Taobao, and a great many more, have turned from being non-existent to becoming an almost essential part of people's daily life. Data about people's offline activities are also captured and stored. Equipped with increasingly powerful (mobile) computing devices, people are not only the consumers of digital services, but also the very contents of them. While they are happily chatting about, posting photos in Online Social Networks (OSNs), purchasing books on Amazon, withdrawing money from an ATM, they ignore the fact that their actions are being recorded, their personal data are being collected, even shared, and the probable consequences of it.

However, because of the sheer amount, complexity and (intentional) obscurity of this data world, people are hindered from understanding it, and leveraging it in order to become more independent individuals and competent citizens. This limitation also presents an opportunity for us. We, as researchers and practitioners, aspire to make tools that help people explore and understand the data collected about themselves, and help them make more informed decisions.

[1] `http://www.slideshare.net/BernardMarr/big-data-25-facts`

Furthermore, when such tools are widely adopted, and data transparency is on everyone's mind, people can be liberated from information monopolization, and break social, economical, political and cultural barriers that spawn injustice, prejudice and inefficiency.

According to Condorcet [43]: "the spread of knowledge through the improvement and democratization of education would contribute directly to political freedom and human happiness". We follow Berendt's [18] proposal to extend Shapiro's and Hughe's [150] notion of information literacy in Section 1.1. We then elaborate on the notions of privacy and knowledge discovery in Section 1.2 and Section 1.3 respectively. We discuss information visualization in relation to the software tools that address privacy and data literacy concerns in Section 1.4. We then give our research questions and contributions in Section 1.5 and outline the thesis in Section 1.6.

1.1 Information Literacy

Following Berendt's approach [18], we first examine the modern notions of Information Literacy (IL) conceptualized by Shapiro and Hughes [150]. Then we look at Berendt's proposal for the extension of the IL curriculum, namely Privacy Literacy and Critical Data Literacy.

In 1996, at the beginning of the fast growth of the Internet, Shapiro and Hughes [150] recognized the inadequacy of the conventional approach to Information Literacy, and proposed a new curriculum, in response to the following questions:

> "What does a person need to know today to be a full-fledged, competent and literate member of the information society?"
>
> or
>
> "What sort of 'information literacy' — an often-used but dangerously ambiguous concept — should we be promoting, and what should it accomplish? Is it merely something that will reduce the number of tech-support calls that we have to deal with? Something that will grease the wheels of the information highway? Something that, as defined by representatives of the library community, enables people to be 'effective information consumers'?"

Shapiro and Hughes argued that Information Literacy should be "something broader, something that enables individuals not only to use information and information technology effectively and adapt to their constant changes but

also to think critically about the entire information enterprise and information society", "Something more akin to a 'liberal art' — knowledge that is part of what it means to be a free person in the present historical context of the dawn of the information age". Shapiro and Hughes then proposed a curriculum with emphasis on what is needed in higher education, which includes:

Tool Literacy the ability to understand and use practical and conceptual tools of current information technology;

Resource Literacy the ability to understand the form, format, location and access methods or information resources;

Social-Structural Literacy knowing that and how information is socially situated and produced;

Research Literacy the ability to understand and use the IT-based tools relevant to the work of today's researcher and scholar;

Publishing Literacy the ability to format and publish research and ideas electronically;

Emerging Technology Literacy the ability to ongoingly adapt to, understand, evaluate and make use of the continually emerging innovations in information technology;

Critical Literacy the ability to evaluate critically the intellectual, human and social strengths and weaknesses, potentials and limits, benefits and costs of information technologies.

Berendt [18] extended the above curriculum with *Privacy Literacy* and *Critical Data Literacy*. *Privacy Literacy* is related to Shapiro and Hughes' *Publishing Literacy* in the sense that people should understand the implications of personal data publishing, and have the "willingness and ability to not publish material on self or others". *Critical Data Literacy* is based on the assumption that "today, the presentation of numbers or other data and their analyses by mathematical-statistical models and/or visualization techniques has become a second very important form of human communication" (in addition to natural languages). Average users should be capable of critically analyzing and understanding aggregated digital data.

This thesis addresses both *Privacy Literacy* and *Critical Data Literacy* concerns. Before we turn to the research questions and the contributions, we first elaborate the notions of privacy and knowledge discovery.

1.2 Notions of Privacy

Privacy has been a major concern over the past decade, and more so when more digital information about individuals is being created, collected and processed, especially when people are involved in various online communities, documenting and sharing their daily lives with others. In the age of online social networking, "even as bloggers and networkers delve into their private experience, they communicate with their fellow humans in a shared festival of the self" [9]. It was found that OSN users had demonstrated high privacy concerns while revealing great amounts of personal information [2]. It also has been quantitatively demonstrated that users' perceptions of audience size do not match reality, since not enough feedback is provided [21]. Boyd showed that collapsed or ambiguous online contexts could lead to undesired disclosure of personal information [29]. People are also familiar with such examples as "Google customizes ads according to your search"[2] or "someone was fired because of his Facebook posts"[3]. The issue is unintended sharing of personal information. Sometimes a person is simply unaware of how his data is collected and processed, by whom, and also unaware of the corresponding consequences.

There have been many efforts addressing privacy problems. For example, Balsa et al. [12] proposed cryptographic solutions that can not only encrypt the messages communicated between entities, but also obfuscate the communication patterns. Sayaf et al. [146] proposed an access control framework that takes contexts and accountability into consideration to help people manage their privacy settings in OSNs. There are also the privacy-preserving data mining methods [175] that prevent inference of undesired patterns while retaining the potential to infer aggregate patterns. However, there is a limitation to the present solution approaches. Although ultimately, privacy is about hiding/showing information, the situations that people need to take into account to preserve privacy are complex and dynamic. A naive take on privacy may prohibit us from negotiating our private well-being.

As Gürses [84] argues, it is insufficient to assume that "privacy is preserved when certain information is hidden", we should establish wider notions of privacy. We examine the notions of privacy from two perspectives: audience type and research approach. Based on the types of audience, there are three notions of privacy [140, 98] that can help us delineate different scopes of privacy concerns[4]:

[2] `https://support.google.com/ads/answer/1634057?hl=en`

[3] `http://time.com/3636220/nordstrom-aaron-hodges/`

[4] Raynes-Goldie [140] differentiated between *social privacy* and *institutional privacy*. For *institutional privacy*, Jameson et al. [98] further made the distinction between privacy concerns raised due to data exploitation by (mainly commercial) institutions, and due to governments' surveillance for political control. We adopt the categorization in [98].

social privacy concerns how to negotiate private-public boundaries with other people, usually peers;

institutional privacy concerns the collection, processing and (re-)purposing of personal data by institutions, mainly commercial entities;

governmental privacy concerns surveillance by governmental entities.

To address these privacy concerns, there are three notions of privacy [85, 84] based on research approaches that can also help us recognize limitations in current research:

Privacy as hiding: Confidentiality — the right to be let alone, the right to a private sphere;

Privacy as control: Information self-determination — the right of an individual to decide what information about himself should be communicated to others, under what circumstances;

Privacy as practice: Identity construction — the freedom from unreasonable constraints on the construction of one's own identity, be it by strategically revealing or concealing data.

The notions of privacy based on research approaches are not mutually exclusive, but emphasize different approaches towards privacy.

Privacy as hiding is the traditional way in which privacy is considered. It bears a sense of intrinsic, protective sphere for an individual to be freed from social intrusions [182, 139]. This formulation, when applied to digitized personal data, implies that keeping one's data completely confident would be the solution to all privacy issues. Applications of encryption/decryption, obfuscation and anonymization are well suited regarding this formulation. However, it does not address the situation where personal data needs to be published, but under control.

Privacy as control defines privacy not only as a matter of concealment of personal information, but also as the ability to control what happens with it. For instance, generally speaking, a person has a right to decide whether to publish his photos and how these photos are used by others. We see this notion appearing in many legal codifications, as elaborated in [19]. It becomes more difficult for an individual to exercise privacy control in the digital space, because data can be much more easily created, copied, modified and disseminated, compared with the offline world. Access control models are typically the solution approaches

regarding this notion of privacy. However, privacy-as-hiding and privacy-as-control merely concern one's private well-being on an individual level: how to hide oneself, how to control the publicized information about oneself. We should not forget that privacy bears a collective connotation — the various public statuses (social, economical, political, etc.) of a group, to which a person belongs, are of profound relevance to that person's privacy, because what should be public or private is often debatable. Public interests affect private affairs and vice versa. An individual needs to have a sense of the public so as to have a better understanding of his own privacy.

Privacy as practice goes beyond control of personal information flow. It defines the perspective of an individual towards privacy as identity construction. This construction requires the individual to be informed about the consequences of his interactions with others, gain awareness about oneself in a broader scope, and when necessary, intervene in the evolution of data aggregation systems. It demands transparency with respect to aggregated datasets, analysis methods and the decisions applied to them [84].

1.3 Notions of Knowledge Discovery

Critical Data Literacy requires people to be able to understand data and discover knowledge from data on a large scale. The ability to perform data analysis and exploration is important to this literacy. Two mainstream models were proposed to capture the whole process of knowledge discovery. They are Knowledge Discovery in Databases (KDD) [57] and Cross Industry Standard Process for Data Mining (CRISP-DM) [152]. Both models show similar divisions of knowledge discovery tasks and both incorporate iterative processes. As shown in Figure 1.1, the selection, preprocessing and transformation phases in KDD correspond to the data-understanding and data-preparation phases in CRISP-DM. Similarly, the data-mining phase corresponds to the modeling phase and the interpretation & evaluation phase corresponds to the evaluation phase. CRISP-DM is more comprehensive than KDD in the sense that it takes the knowledge discovery process in its full application context. We will be using the terms in CRISP-DM henceforth.

Conventional data analysis tools serve data mining experts, business analysts and practitioners who are responsible for designing and maintaining intelligent systems. They do not nourish a "data-understanding" environment for non-experts. An average user does not necessarily investigate each phase of the knowledge discovery cycle to gain insight in data, nor should he/she. The reasonable discovery phases in which a user should be involved are data-

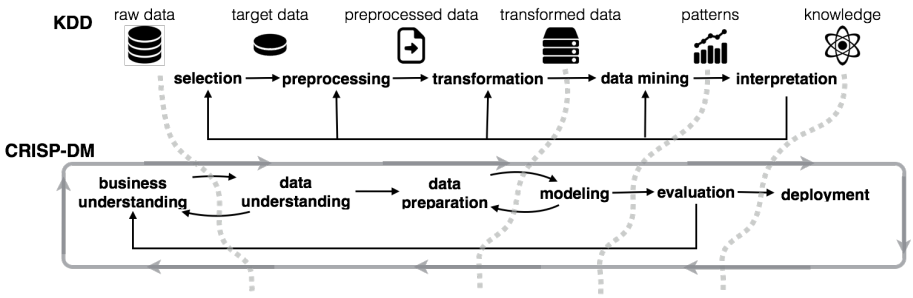

Figure 1.1: The KDD model and the CRISP-DM model that summarize the process of knowledge discovery.

understanding, evaluation and deployment. These aspects are also closely related to privacy-as-practice, because the transparency towards aggregated data sets about personal information enables a person to adjust the boundaries of his private space, constructs his public identity, and perhaps more importantly, contributes to the public good. When everyone is more equipped with the ability to understand the data relevant to him/herself, there is an increased chance to counter the information imbalance between powerful entities such as governments, corporations, and the individuals who are often in vulnerable positions. As discussed in [19], the exploration, selection, integration, construction and modeling of aggregated data can easily result in privacy violations, for linkages can be made that lead to the identification of individuals, or filtering is conducted based on potentially discriminatory criteria. This compromises data transparency[5]. Platforms need to be built to promote aggregated data transparency, to help a user overview a certain data ecosystem, of which he is a part, reflect on his and possibly his peers' past actions, re-define his privacy boundaries and make more informed decisions in the future.

1.4 Feedback and Awareness Tools

To promote privacy-as-practice for *Privacy Literacy* and data-understanding, evaluation and deployment for *Critical Data Literacy*, we need a a new category of tools, namely Feedback and Awareness (FA) tools [17]. These tools are essentially data analysis and exploration tools. But they go beyond traditional tools in the sense that they are designed for and used by non-experts. They

[5]Data transparency may have different meanings in different contexts. In this thesis, by data transparency, we mean the accessibility of the data in a system or a process for the person, who is using the system or involved in the process, to understand the data.

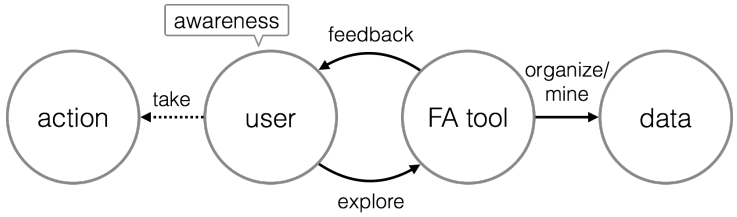

Figure 1.2: The iterative process in which a user interacts with the data and gains awareness, in order to take action.

are also different from other "dashboard" tools that show static info-graphics or conventional information retrieval tools that focus on search. Instead, they emphasize more on data exploration. They should be intuitive to use, but versatile and flexible enough so as to enable and encourage user exploration, and lead to potentially various insightful discoveries. Because of the rich feature set or functions provided by an FA tool, the user can interact with it iteratively. In this process, the user (gradually) gains awareness, and learns new knowledge. Eventually, the user may feel competent to take action. Or the awareness is an end in itself. This process is illustrated in Figure 1.2.

In the phase of data-understanding, an FA tool collects, describes and presents data in a clear and interactive way and allows further exploration and manipulation of the target data. In the phase of evaluation, an FA tool performs data-preparation and data-modeling on behalf of its users and hides computation details. The users can then evaluate the feedback from the tool and make informed decisions to an improved extent in the phase of evaluation. The tool can offer ways to realize the users' decisions in the deployment phase. When FA tools are applied to aggregated personal data, they not only help users make privacy-related decisions, but more importantly, provide an informative and usable ecosystem that enlightens people and encourages them to reflect on their "digital well-being".

Information visualization plays an important role in FA tools. As Hutchins [97] pointed out, intellectual work is accomplished as a kind of interaction with cognitive tools — pencils and paper, calculators, and, increasingly, computer-based intellectual supports and information systems [180]. With the advancement of computer hardware and software, information visualization plays a more important role in such cognitive tools. According to Ware [180], visualization has the following advantages:

- Visualization allows people to comprehend huge amounts of data;

- Visualization allows the perception of unanticipated, emergent properties;
- Visualization makes errors and artifacts in the data apparent;
- Visualization facilitates understanding of both large-scale and small-scale features of the data;
- Visualization facilitates hypothesis formation.

These advantages have made visualization a powerful way to conduct data exploration. Information visualization is the study of (interactive) visual representations of abstract data to help people interpret and comprehend data. It is a subfield of computer-based visualization. Information visualization offers a supplement in emphasizing the importance of giving users an overview and insight into the data distributions. Traditional statistical methods are good at hypothesis testing, but bad at data exploration without a given hypothesis. Information visualization harnesses human visual abilities, making data exploration its strength. Shneiderman [155] argues that combining statistical methods and information visualization leads to novel discovery tools that preserve user control and enable more effective exploration. Therefore, in this thesis, we mainly study the exploratory visualization design options towards online privacy and data literacy problems.

1.5 Goal, Questions and Contributions

The overall goal of this thesis is to develop solutions and build exploratory visualization tools to address the problems in online social privacy and aggregate data transparency.

We differentiate between two levels of our research focus: the micro-level and the macro-level. On the micro-level, we address social privacy concerns for individual OSN users in their egocentric networks. On this level, the user of an exploratory visualization tool only deals with his/her own ego-network. On the macro-level, we look for solutions that reveal patterns of aggregated data beyond online ego-networks. We have the following research questions:

1. How do we address OSN users' social privacy concerns on a micro-level?
2. How do we promote data transparency for aggregated datasets on a macro-level?
3. How do we design exploratory visualization tools with programming libraries for interactive data visualization?

The corresponding contributions are summarized as follows:

- We developed a friend-grouping-and-exploration tool, named FreeBu, for Facebook users. It addressed social privacy issues within ego-networks on Facebook via the privacy-as-control and privacy-as-practice approaches. It also addressed critical data literacy in terms of data-exploration, evaluation and deployment.
- We conducted the user study that investigated which algorithm could be more useful for Facebook users to construct target friend groups: hierarchical, modularity-based community detection used in an interactive setting (HMOD), or Facebook smart lists (FSL). We found that the former is more useful for the users.
- In the user studies in collaboration with SMIT, VUB, we found that the main affordances of FreeBu are friend-removal, friend-grouping, overview and reflection. We also analyzed the user interaction data, identifying which visualizations the users favored in terms of mouse interactions. We then linked the detailed interaction patterns to future visualization design.
- We studied how the friend groups produced by community detection methods matched users' manually created friend groups. We discovered that the modularity-based method (with only the friend-graph as input) produced more accurate groups than others. We proposed an extension to the modularity-based community detection method. Compared with other community detection methods, this extension allows the overlapping communities that better match OSN users' manually created communities in three ego-network datasets.
- We studied how user sentiment expressed in online posts and chats differed among subgroups. We designed and implemented two algorithms to extract comparisons of subgroups in terms of categorical item sets, towards one numerical target variable. Unlike traditional subgroup discovery that looks for interesting subgroups distinct from the entire population, these algorithms aim to find comparisons between subgroups locally. We identified interesting subgroup comparisons, and make connections to existing sociological studies.
- We developed a discrimination-rule-exploration tool, named D-Explorer. This tool was designed for examining and evaluating the association or classification rules produced by discrimination-aware data mining algorithms, and aiding in new hypothesis formation. A "bank-loan decisions" (offline activities) dataset was used for demonstration, but the tool could be readily extended to other data domains. This approach

addressed critical data literacy in terms of data-exploration, evaluation and deployment. Also, as D-Explorer informs the user about the consequences from the aggregation of the user's personal information, the tools helps user with regards to institutional privacy in terms of privacy-as-practice.

- We surveyed the current generation of online data-visualization libraries. We developed a framework for evaluating and designing such libraries and proposed future directions for improvement.

1.6 Thesis Outline

We outline this thesis as follows:

Related work

Chapter 2: *Related Work* — We review existing solution approaches towards the concerns regarding online privacy and data literacy. We also look into the related work on library design for interactive online data visualization.

Part I : Online Social Privacy

Chapter 3: *Addressing Context Collision* — A first step towards addressing the "context collision" problem in OSN: we answer such questions as "How does a person categorize the people he/she knows?" and "How do we assist a user in making such categorizations?", and the prototype FreeBu#1 is developed. (publications: [69], [45])

Chapter 4: *Using Friend Groups to Avoid Regretted Posts* — The hierarchical modularity-based community detection algorithm is compared with Facebook smart lists for avoiding regrets, with the improved interface, FreeBu#2. (publication: [66])

Chapter 5: *Investigating Community Detection in Ego Networks* — We investigate how the friend groups produced by the modularity-based community detection algorithm match OSN users' manual groups, and propose an improvement that allows overlapping groups. (publication: [67])

Chapter 6: *Extending and Evaluating FreeBu* — FreeBu is extended to include more views on friend-grouping (FreeBu#3). We also investigate how users perceive the values of the tool, and how they interact with it. (publications: [67], [45])

Part II : Transparency in Aggregated Data

Chapter 7: *Comparing Patterns in Social-sentiment Mining* — We test and mine OSN users' sentiment differences among demographic and privacy groups, and propose new algorithms to discover different groups in terms of sentiment expression with multiple attributes. (publication: [70])

Chapter 8: *Rediscovering Patterns in Discrimination-aware Mining* — We develop a visualization tool, namely D-Explorer, to help users discover sensitive profile attributes or combinations of them that could potentially lead to discrimination. The tool is based on the discrimination-aware data mining system DCUBE, and operates on a "bank-loan decisions" dataset. (publication: [65])

Part III : Practical Visualization Design

Chapter 9: *Investigating Online Data-visualization Libraries* — We conduct comprehensive evaluation and comparison of state-of-the-art online data-visualization libraries. We examine the libraries in terms of application domains, abstraction levels, visualization tasks and design patterns. We also propose library design improvements for visualization development. (submitted: [68])

Chapter 10: *Redesigning FreeBu and D-Explorer* — Informed by previous studies, we redesigned and re-implemented the two exploratory data visualization tools: FreeBu and D-Explorer.

Conclusion

Chapter 11: *Conclusion* — We summarize the thesis and take an outlook on future work.

Chapter 2

Related Work

In addressing the research questions listed in Section 1.5 of Chapter 1, we review the existing approaches and establish connections between the related work and the focus of this thesis.[1] We categorize the existing approaches into two categories:

- Approaches towards online privacy
- Approaches towards aggregate data transparency
- Approaches towards data-visualization library design

2.1 Approaches towards Online Privacy

The first category of related work emphasizes the protection of people's privacy when they conduct online activities. As introduced in Chapter 1, there are three research approaches towards understanding the concept of privacy and recognizing corresponding privacy concerns more comprehensively. They are privacy-as-hiding, privacy-as-control and privacy-as-practice. We organize the related work towards online privacy in the computer science according to these three research approaches.

[1] This chapter is closely baed on the projection deliverable [3]. It has been updated and extended by Bo Gao.

1. privacy as hiding — cryptographic solutions that not only hide users' communication from the social network service provider (such as Facebook), but also hide communication from non-participating peers;
2. privacy as control — access control models that incorporate rules or settings (such as the current OSN privacy settings) to govern the information flow within a user's social network;
3. privacy as practice — Feedback and Awareness (FA) tools that provide data-understanding-and-exploration utilities to empower users, and improve existing OSN contact management options (such as Facebook friend lists).

The three research approaches supplement each other and form a synergistic solution ecosystem. The limitations of the first two approaches are as follows:

Cryptographic solutions encrypt messages that only the designated receivers can decrypt. This introduces an extra layer of encryption-key exchange mechanism, increases the level of difficulty and does lead to the confusion of users [11]. It also goes against the current economic model of OSNs that relies on the access to users' personal data to benefit from advertising. Cryptographic tools block this access, making such tools economically difficult to be widely implemented on OSN platforms.

Access control models (e.g. [145, 146]) provide elaborated privacy options and automate certain ruling procedures for deciding the target audience of online posting. These approaches, compared to the current OSN implementations, make the underlying privacy settings more transparent and empower users with more control. However, users are still the decision makers who evaluate what, when, how to post online, and to whom. To make such decisions, users need not only access control, but also an overview and an understanding of their own social network data. This leads to FA tools that take the privacy-as-practice approach to fill this gap. We focus on the FA approach in this thesis.

Berendt [18] described FA tools as "[taking] the advantage of the liking that many people have to 'look into a mirror', where the phenomenon being studied is — them". In the case of OSNs, this description can be extended to apply to not only the data from the OSN user himself, but also the data from the egocentric network that the user created and has maintained. A related concept is "nudging" or "soft paternalism", introduced by Acquisti et al. [83]. The "nudging" tools usually introduce extra mechanisms to alert users before they take actions, such as a timer that delays the user's submission of his post.

FA tools do not explicitly nudge users into reconsideration of their actions. Rather, FA tools present different aspects of the user's data, enable data-

exploration and aim at fostering understanding and reflection. For example, to address privacy issues, Lederer et al. [112] suggest improving privacy sensitivity in systems through feedback that enhances users' understanding of the privacy implications of their system use. This can be coupled with control mechanisms that allow users to conduct socially meaningful actions through them. These ideas have led to suggestions like the identityMirror [117], which learn and visualize a dynamic model of user's identity and tastes. Similar ideas are embodied in the concept of privacy mirrors [135] or in the proposal for linkage control in identity management systems [87].

2.1.1 Privacy-relevant Feedback and Awareness Tools

Privacy-relevant Feedback and Awareness (PFA) tools help range from simple privacy-setting checklists to complex exploratory contact-management tools that aid users in grouping their online contacts and exploring various attributes and connections of their friends. Through these tools, the user can be informed about the current configuration of their privacy settings. With these tools, users can also at their social networks from different perspectives, derive new knowledge and reflect upon the relationships that the users have with their online contacts.

In the this subsection, we present a selection of existing PFA tools.[2] The search strategy for compiling the tools to be included in the survey was to use a general search engine (Google) as well as a major social-networking site (Facebook) and two special-purpose websites: Information Aesthetics[3] and Visual Complexity[4]. Information Aesthetics is a weblog that explores the symbiotic relationship between creative design and the field of information visualization. More specifically, it collects projects that represent data or information in original or intriguing ways. VisualComplexity.com intends to be a unified resource space for anyone interested in the visualization of complex networks. The website's main goal is to leverage a critical understanding of different visualization methods, across a series of disciplines, as diverse as Biology, Social Networks or the World Wide Web.

On all sites, search was done via the keywords privacy, feedback, awareness, tool, social network, web, and visualization. The list of applications was filtered by a number of criteria to select a number of applications that are as "representative" as possible, yet can be described in the limited available space. Criteria such

[2]The content of this selection is mainly from the our survey on privacy-relevant FA tools in the technical report [3].

[3]`infosthetics.com/`

[4]`www.visualcomplexity.com/vc/`

as the environment (e.g. SNS, blogosphere, the Web in general, etc.), the data being analyzed (e.g. profile data, phone records, blog texts/comments, browser cookies, etc.) or the people being analyzed (e.g. a single user, several users, a specific web social media community, everyone on the Internet, etc.) are taken into account when selecting the tools, and we try to make the selection diverse and representative. Aesthetics and usability of an application is also taken into account. With the same functionality, more "beautiful" and "usable" applications will be chosen. This search strategy necessarily has subjective components and cannot claim completeness of all existing applications (no search engine has full coverage of the Web), but it did produce a good overview.

However, a wide sense of the terms "privacy-relevant feedback and awareness tools" could comprise any tool that shows any information about some data that could be privacy-relevant. This could even be a search engine (because it allows one to search for blogs in which people talk about personal matters), a news aggregator (because it may condense privacy-relevant information and thus lead to new awareness of it) or Facebook itself (because it makes one's data behaviour explicit, which may lead to new awareness). A narrow sense of the term could comprise only tools that give feedback about one's so-called "privacy settings" in Facebook. We do believe that it is important to also look into the wider meaning of the term FA tools in order to do justice to the notion of "privacy as practice", which calls for an open investigation of practices of data creation, hiding and revealing. However, we also want to structure the term in order to not let the term become too vague. We therefore decided to classify tool candidates by their purposes (privacy role), with some aiming at detecting and revealing privacy breaches, and others at mirroring or summarizing privacy-relevant information for the user.[5] We refer to the classification of [85], which identifies the following privacy breaches[6]:

1. Indeterminate Visibility: Indeterminate visibility denotes the problem of a user's profile information being visible to others without the user's explicit knowledge or approval.

2. Aggregation of Separated Digital Identities: Separation of digital identities denotes the construction of social identities by individuals that selectively reveal and conceal information in specific contexts and roles.

[5] These purposes are not mutually exclusive.

[6] Note that this classification was originally intended to describe privacy breaches in OSNs. However, it can be readily extended to the context of the entire web. Also, the fourth breach type identified in the classification relates to Contested Ownership: the explicit and implicit definitions of data ownership that may lead to privacy breaches. Tools dealing with this notion were not found here.

3. Misappropriation: the use of users' personal data out of context or for previously undefined purposes.

Interestingly, several tools show that in order to think about problems such as visibility at all, one has to have or get an overview of "what one has done". We call such tools

4. Mirroring: giving the user an overview of their data and/or actions.

In addition, we categorized the selected applications based on the *time* criterion. The categorization based on time assumes that a person has initiated or participated or is simply interested in a series of events by him/her-self on the web through time. We call this collection of events associated with this user the activity stream [93]. These events may include changes that the user made to their profile page, the fact that the user added or ran a particular application on the social networking site, that they shared a news item, or that they communicated with one of their friends or commented in a forum, etc. Now that we have the axis of time, we are able to distinguish between the past and the future. There are PFA tools that summarizes the history of a user using the web and presents the resulting statistics (regardless of what kind of form, plain texts or graphics) to the user. We call these tools Historical Viewers. The other category of PFA tools are the ones that show the consequences of a user's current actions or actions by others that are relevant to the user as well as predictions for the future (using e.g. what-if simulation). We call these tools Ongoing Monitors. We can see that the data for Historical Viewer to process are static since they are historical facts, while the data fed to Ongoing Monitors are dynamic, because of their ongoing nature.

Indeterminate Visibility

These tools give feedback on the visibilities of one's historical data or of one's historical/current settings – which may have implications for future data, or help users manage the visibilities of their data.

• *Privacy Check*[7] – it shows how much profile information third-party applications can access. The tool is a Facebook app, which uses standard Facebook API to check the data items' visibility in user's profile. It also calculates a general score to measure the status of the user's profile privacy settings.

[7] `www.rabidgremlin.com/fbprivacy/`

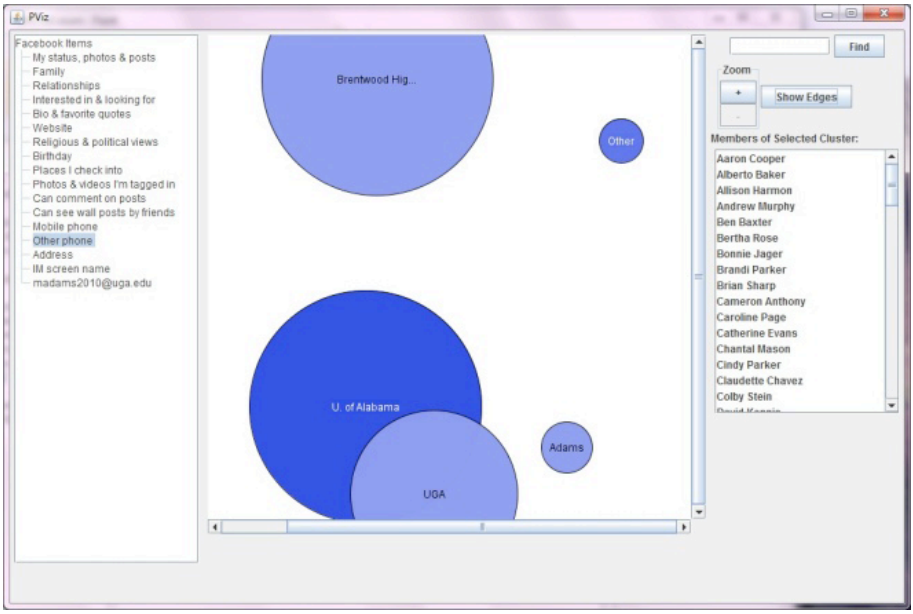

Figure 2.1: Coarse granularity view in PViz.

• *PViz* – As stated in [123], "PViz is an interface and system that corresponds directly with the way users model groups and privacy policies applied to their networks. It allows the user to understand the visibility of her profile according to natural sub-groupings of friends, and at different levels of granularity. PViz is centered on a graphical display, which shows the user's social network. Each node in the display represents a semantically meaningful sub-group of the user's friends (a community) or an individual friend. Figure 2.1 shows a screenshot of PViz displaying Margaret's social network. Inspecting the display shows that PViz has found five main communities of friends".

• *Privacy Wizard* [55] – Privacy Wizard is a fine-grained tool that helps the user configure which information is hidden from/ revealed to which friend semi-automatically, which means that the user only has to configure a subset of the privacy settings manually, and the wizard will use its underlying machine learning model (a classifier) to automatically configure the rest. Two screenshots are shown in Figure 2.2. With an increasing number of friends that the user has manually checked, the machine learning model becomes more and more confident in recommending to the user good privacy configurations for the rest of her friends.

Figure 2.2: Privacy Wizard learns through the configuration of privacy items for user's friends. It becomes more confident when more friends are configured.

• *deGeo*[8]– A growing number of tools exploit the fact that publishing location data in any way may allow for sensitive inference. *Please Rob Me*[9] or *I Can Stalk You*[10] demonstrates how to automatically identify empty homes (on the basis of information in Tweets) or people's whereabouts (on the basis of uploaded photos' meta data). There are also protection tools built on these ideas, *deGeo* is a photo sharing privacy tool for iOS[11] devices, of which the users can share photos without compromising locational privacy. deGeo works by removing the embedded geotags as well as associated EXIF[12] metadata before sharing photos online or with friends since an iOS camera app stores the exact GPS location with each photo that a user takes.

Aggregation of Separated Digital Identities

The following tools take two very different approaches to the separation of identities. The first focuses on the de-separation of virtual identities as a feature, a desired simplification of a multitude of virtual identities in various social networks[13]. However, it could be used also as a feedback tool on the degree to which there is undesired de-separation. The second focuses on a user's belief to be "hidden in a crowd" and thus able to separate their browsing identity from their "real-life, unique identity". Both tools are based on historical data or settings and thus are Historical Viewers.

• *About Me* – As aptly described in its Wikipedia page[14]:"The site offers registered users a simple platform from which to link multiple online identities,

[8] `http://itunes.apple.com/us/app/degeo-photo-sharing/id412472011?mt=8`.
[9] `http://pleaserobme.com/`
[10] `http://icanstalku.com/`
[11] `http://en.wikipedia.org/wiki/IOS_(Apple)`
[12] The specification for the EXIF format can be found at `www.cipa.jp/english/hyoujunka/kikaku/pdf/DC-008-2010_E.pdf`
[13] Another similar site is *ClaimID* (`http://claimid.com/`).
[14] `https://en.wikipedia.org/wiki/About.me`

relevant external sites, and popular social networking websites such as Facebook, Flickr, Google+, Pinterest, LinkedIn, Twitter, Tumblr, and YouTube. It is characterized by its one-page user profiles, each with a large, often artistic background image and abbreviated biography."

• *Panopticlick*[15] – Panopticlick [53] tests user's browser to see how unique it is based on the information (such information includes browser type, browser plugin details, time zone, screen size, whether cookie is enabled) it will share with sites it visits. Such information is also called a "browser fingerprint". The user is given a uniqueness score (in bits) calculated based on the fingerprint. The higher the score is, the more identifiable the user becomes. Note that we put Panopticlick into the Ongoing Monitor category because it always measures user's current browser fingerprint. However, user's past browser fingerprint can be logged using this tool and become a fingerprint history of the user on the web.

Misappropriation

Many misappropriate uses of online personal data exist. Companies or organizations may unobtrusively or secretly repurpose users' data. But this is technically difficult, or impossible to detect. Most of the time, educational materials are used to help people realize potentially privacy-harmful situations. Lightbeam is an online application that informs the user about the potentially misappropriate usage of their browsing histories:

• *Lightbeam*[16] – The idea is that when visiting website A, other websites (e.g. B and C) are possibly tracking the user's browsing history, clicked items and at the same time, collecting the user's browser and computer info through A. This collected information is likely to be sufficient to uniquely identify the user. Lightbeam a Firefox add-on that interactively visualizes which third-party data collector track users with browser cookies when they use the web. Figure 2.3 shows that when visiting the imdb.com web site, several other websites are revealed to be trackers such as Google and Amazon. The circle in the middle is the site visited by the user. The triangles connected to the circle are the trackers. The graph is extended as the user continues browsing. 2.4 shows a detailed list of trackers and visited sites. The user can block the tracking by third parties with this tool. However, the tool does not provide the information on which website is tracking which specific information.

[15] More information can be found on the website: `https://panopticlick.eff.org/`.
[16] `https://addons.mozilla.org/nl/firefox/addon/lightbeam/`

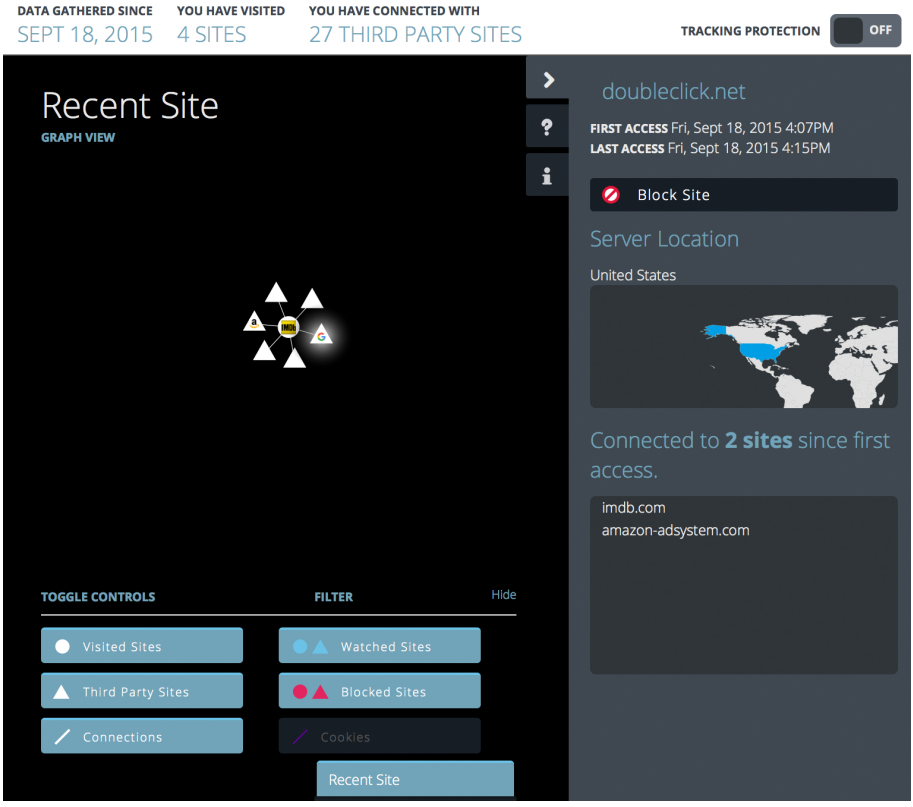

Figure 2.3: Lightbeam graph view

• *DataBait*[17] – DataBait is a tool set that shows the user who tracks him/her on the internet" and the predictions (e.g. personality) that can be derived by analyzing the user's digital trail on the web. It is also claimed that DataBait "gives an indication of the economic value of your profile and your Facebook friends". DataBait is a deliverable under the European research project USEMP (user empowerment for enhanced online management)[18]. DataBait currently includes a Facebook application and a browser plugin. However, both tools are still under development and not available to the public.

[17] `https://databait.hwcomms.com/`
[18] `http://www.usemp-project.eu/`

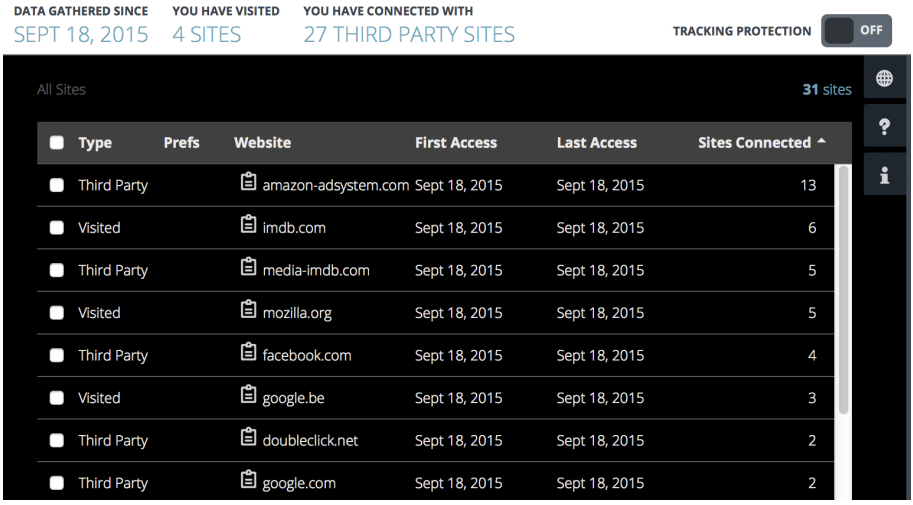

Figure 2.4: Lightbeam list view

Mirroring

Based on current and/or the user's historical data and settings, the following Historical Viewers (*Personal Analytics*, *Social Memories* and *Tell-All telephone*) and the Ongoing Monitor (*PRISM*) give information "about me": what have I done or what am I doing? *coComment* is a mixture in the dimensions of *time* and *space*, i.e. it runs on both current and historical data about both "me" and "the world".

• *Personal Analytics for Facebook*[19] – As part of the Wolfram Alpha knowledge Engine[20], it is a visual and textual analytic web application designed for Facebook users. It offers a wide range of analytics, including various friends' demographic reports, summaries of the user's log-ins, posting and sharing activities, etc. Another merit of this tool is that each analytic segment can be downloaded in different formats for other uses, such as a spread sheet, image and vector graph.

• *Social Memories*[21] – A collection of Social Memory figures is automatically generated from user's Facebook data, including the user's profile data, album photos, status updates, friends' profiles and messages, etc. Various infographics

[19] `www.wolframalpha.com/input/?i=facebook+report`
[20] `www.wolframalpha.com`
[21] The application can be found at `www.facebook.com/socialmemories`.

Figure 2.5: Screen shots from Social Memories, including "friend gender distribution", "your biggest (photo) album", "your vocabulary" and "most active friends".

are used to illustrate data trends or patterns, e.g. the most popular photo album, friends most tagged with the user's name, who the user photographed the most, distribution of friends, most active friends, most popular tags, friend gender distribution, status vs. responses, the events the user attended, weekly activity statistics, friend home towns, and so on. All the historical data in the user's account is used. This is an excellent application in the sense that it provides the user a fairly comprehensive overview of one's history on Facebook through clear, precise and intuitive infographics, in which users can learn their own past behaviors, their friends' responses, etc. Several screenshots show how this application looks in Figure 2.5.

• *PRISM* [137]– is short for PRIvacy-Sensitive Messaging system, a plugin for an open-source Instant Messaging system. It provides IM users with various visualizations that allow for greater visibility (to oneself) of one's own actions in relation to one's contacts (e.g., temporal patterns of login activity, periods of idleness). PRISM provides mechanisms for presenting oneself differently to various groups of contacts by selecting different impression-relevant settings, such as "online", "away" or "appear offline", etc., for them.

• *Tell – all telephone*[22] – German politician Malte Spitz sued to have German telecoms giant Deutsche Telekom hand over six months of his phone data that he then made available to ZEIT ONLINE, who combined this geo-location data

[22]Note that *Tell-all telephone* is strictly speaking not a PFA tool, rather, a project or demo, since it only renders one particular person's geo-location data. However, we think the idea behind it is valid and useful for building a geo-location awareness privacy tool.

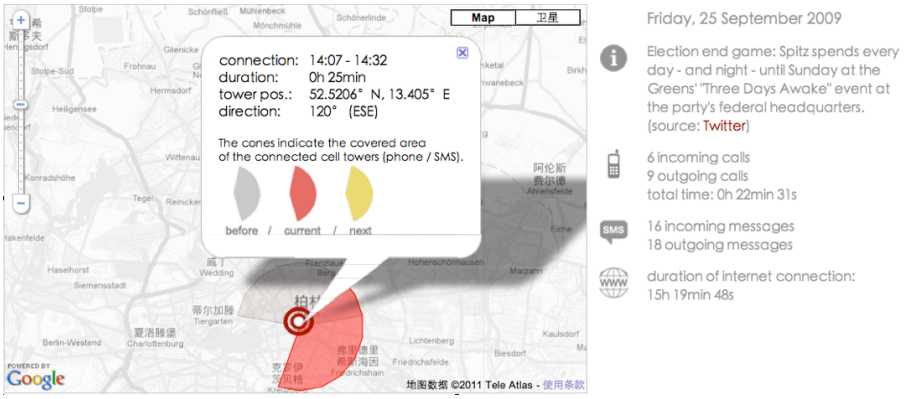

Figure 2.6: Tell-all Telephone

with information relating to his life as a politician, such as Twitter feeds, blog entries and websites, all of which is all freely available on the internet. This results in a privacy awareness tool[23] specifically showing the mostly private activities of Malte Spitz from August 2009 to February 2010. From Figure 2.6 we see that a surprisingly large amount of information can be revealed through the phone log files, such as the number of phone calls/SMS messages, duration of Internet connection, the route of the user with time stamps, even the location and the coverage of the tower. Although the purpose of such tool is to support Malte Spitz's point that revealing phone logs is harmful to personal privacy, we can consider it as an augmented geo-location informer that collects user's geo-location information on the web, and combines other public data of user, such as tweets and blogs to raise the user's privacy awareness.

2.1.2 Discussion

Some FA tools focus on institutional privacy or governmental privacy, where users gain knowledge about the ways personal information could be collected and misused by third-party companies or other organizations. These tools include the ones in the "Aggregation of Digital Identities" and "Misappropriation" categories, and also Tell-all Telephone.

The majority of the tools relate to social privacy in OSNs. Typical ones are in the "Indeterminate Visibility" category. However, these tools follow the privacy-as-control approach. Some check and alert OSN users on what information is visible, to which extent, such as Privacy Check, PViz, etc. Privacy Wizard

[23] www.zeit.de/datenschutz/malte-spitz-data-retention

goes on step further in helping users semi-automatically adjust their privacy settings. Only a few follow the privacy-as-practice approach by providing users with overviews of various aspects of their egocentric networks. These tools are Personal Analytics for Facebook and Social Memories. Both only offer static information graphics without interactivity. This limits users' choices in exploring their own networks. Furthermore, there is a lack of connection between privacy-as-control and privacy-as-practice. Once the user explores and gains new insight about his social network, he may want to take certain actions to control his personal information flow.

In this thesis, we develop a tool named FreeBu that helps OSN users both explore their egocentric networks and control information flow. More detailed related work will be discussed in Chapter 3, 4, 5, 6 amd 10 respectively.

2.2 Approaches towards Aggregate Data Transparency

The second category of related work includes a wide range of tools that promote aggregate data transparency, which is an approach that makes data more accessible and understandable. Aggregate data transparency addresses the critical data literacy aspect in the information literacy curriculum. It is especially relevant when the data about the individual is also part of the data that is being aggregated and mined. Sometimes biased, even discriminatory decisions may be made on a collective level with aggregated datasets, and affect certain groups of people. A typical example is when banks use data mining models to help determine whether to approve or reject a loan to someone. Based on historical records, a model may misuse the fact that often females were rejected, and tune its parameters to discriminate against females in general. Another example is when the data about OSN users are aggregated and different services, such as advertisements and recommended news, are tailored for targeted user groups. People have a right to know how their data can be used collectively, what may be the consequences resulting from the inference or mining on the aggregated data. They should also be able to play a constructive role in such decision processes. Data transparency tools can help them, including FA tools.

There is a spectrum of FA tools that can be differentiated in terms of the extent to which they offer data visualization. Recall from the previous section, there are the ones that are with little visualization, e.g. Privacy Check. Such tools primarily rely on texts and numbers to convey information, not the visual positioning, coloring and transitioning, etc. that are typical in data visualization. There are also the ones that incorporate static data visualization,

i.e. users can not interact with the visualizations, such as Personal Analytics for Facebook. Then there are the tools that incorporate basic interactive visualization components, such as PViz, in which the user can limitedly interact with the visual objects for one or two variables in a single view. Finally, there are the tools that rely on visualizations with rich interactivity for users to explore and understand data, for several or more variables, with multiple views. We call them exploratory visualization tools.

In this thesis, we focus on two use cases — discrimination-aware visual mining and OSN user sentiment comparison. More specifically, in the use case, we developed an online exploratory visualization tool named D-explorer that helps users explore mined patterns on potential discrimination, as elaborated in Chapter 8 and 10. In the second use case, we tested social hypotheses about sentiment expression on OSN data and developed algorithms to extract comparisons of OSN-user subgroups that are different in sentiment expression. This work serves as a first step towards building exploratory visualization tools that aid OSN users in gaining insights in their networks on a macro-level, as elaborated in Chapter 7. Next, we review the related work on visualization tools for data analytics and exploration.

2.2.1 Visualization Tools for Data Analytics and Exploration

All the tools mentioned in this subsection provide standard data reporting utilities, including tabular and geographical data processing and conventional visualizations such as bar chart, treemap, geo-chart. They do not address our use cases in terms of parsing classification rules, associating rule items, measuring meta-level characteristics of discrimination rules, comparing sentiment OSN-user groups and visualizing patterns in our use cases with tailored visualizations (as detailed in Chapter 8, 7 and 10). However, through this survey, we gain a clearer view of the current generation of exploratory visualization tools, and can make more informed decisions in developing our own tools.

Commercial Business-intelligence Applications

Business Intelligence (BI) is a term frequently used in industry to refer to the techniques or tools that "[transform] raw data into meaningful and useful information for business analysis purposes"[24]. There exist plenty of

[24] `https://en.wikipedia.org/wiki/Business_intelligence`

BI applications, such as Tableau[25], QlikView[26], BIME[27], Jaspersoft[28], Metric Insights[29]. These tools require a purchase to be used unlimitedly and are mostly desktop tools that need installations.

Visualization Packages in Desktop Computing Environments

There are also the visualization packages that come with computing environments/languages such as R[30], MATLAB[31] and Python[32]. However, in order to process and visualize data, users have to install the corresponding environments and learn the full-fledged languages. The exploration of data relies on the user to type in commands or write complex programs. The variety and the interactivity of the visualizations provided by these packages are also more limited than those of BI applications.

Online Public Data Explorers

Online public data explorers are freely accessible websites, often equipped with rich visualization templates that are usually more intuitive to use than command lines. Users can explore public datasets and/or upload their own to inspect. Typical examples of general-purpose public data explorers are Google Public Data Explorer[33] and RAW[34]. There are also special-purpose ones such as We Feel Fine[35]. We consider these tools good examples for us to follow and build our own tools.

Google Public Data Explorer provides a series of interactive data visualizations for people to explore a wide range of public datasets from various international organizations and academic institutions. These datasets mostly contain tabular data, sometimes geographical data. Google created a new data format DSPL (Dataset Publishing Language[36]) so that anyone can upload, visualize and share their own datasets using Google Public Data Explorer. A screenshot of the tool is shown in Figure 2.7. The user can select an dataset from its data repertoire

[25] http://www.tableau.com/
[26] http://www.qlik.com/
[27] https://www.bimeanalytics.com/
[28] https://www.jaspersoft.com/
[29] http://www.metricinsights.com/about/
[30] https://www.r-project.org/
[31] http://nl.mathworks.com/products/matlab/
[32] https://www.python.org/
[33] http://www.google.com/publicdata/directory
[34] http://raw.densitydesign.org/
[35] http://wefeelfine.org/
[36] https://developers.google.com/public-data/?hl=en

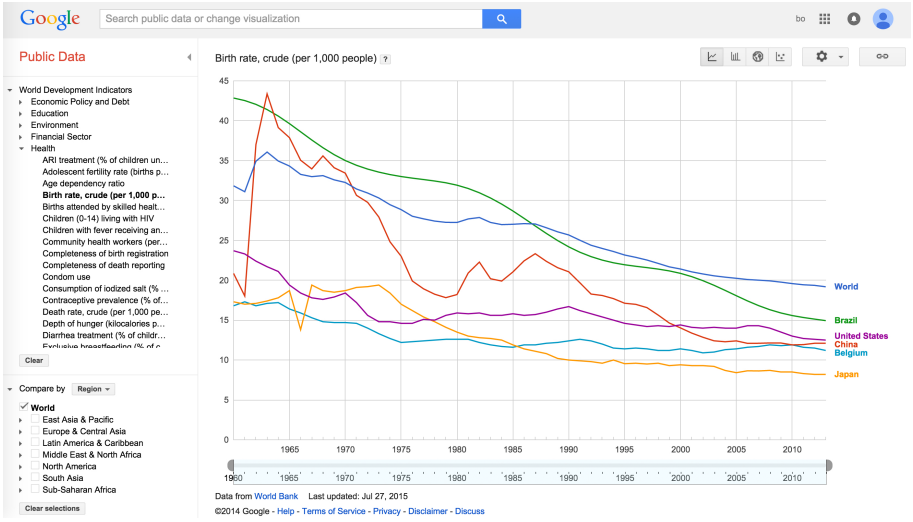

Figure 2.7: A screenshot of Google Public Data Explorer

in the data-selection panel on the left, for example, a dataset on birth rate. The user can then filter the regions or countries shown in the visualization by checking the checkboxes on the left. The user can perform more detailed filtering and highlighting by interacting with the visualization on the right. Different views are available as well. Figure 2.7 shows a multiple-line chart, the views with bar chart and bubble chart are available at the top right corner of the visualization. This tool is valuable in the sense that:

1. It has a rich data repertoire that documents some of the key human developments;
2. It provide multiple visualization views with rich interactive functions.
3. It is a generic data visualization tool that allows rendering custom data.
4. It is freely and openly available.

We Feel Fine [100], as stated on the website of We Feel Fine project[37], "We Feel Fine has been harvesting human feelings from a large number of weblogs. At the core, We Feel Fine is a data collection engine that automatically scours the Internet every ten minutes, harvesting human feelings from a large number of blogs. Blog data comes from a variety of online sources, including LiveJournal,

[37] `www.wefeelfine.org/`

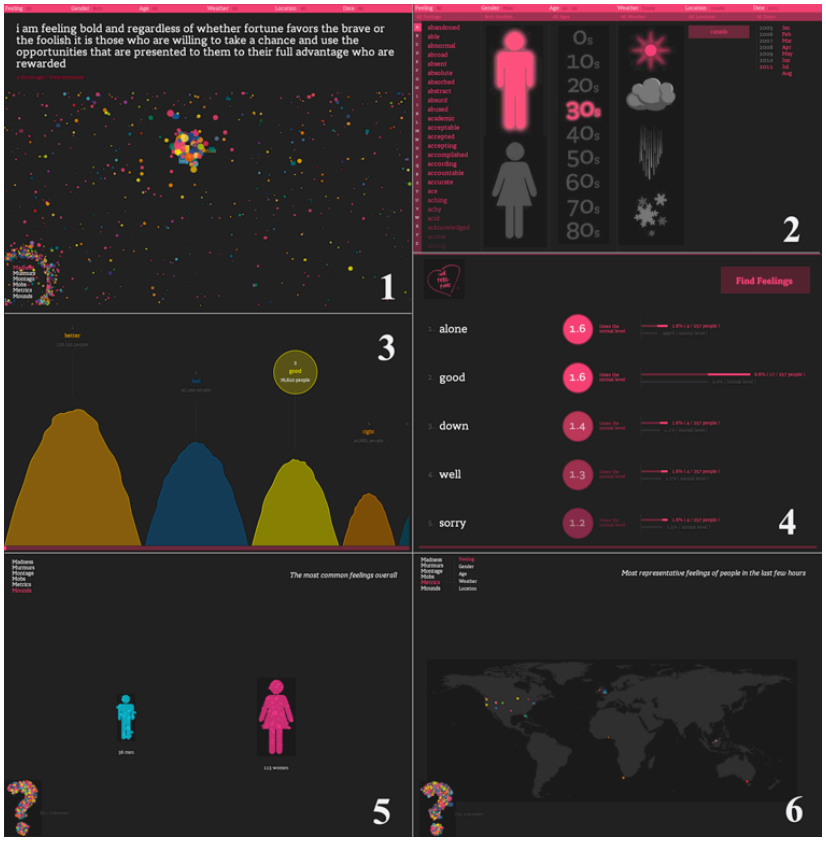

Figure 2.8: screenshots from We Feel Fine

MSN Spaces, MySpace, Blogger, Flickr, Technorati, Feedster, Ice Rocket, and Google". The system searches the world's newly posted blog entries for occurrences of the phrases "I feel" and "I am feeling". When it finds such a phrase, it records the full sentence, up to the period, and identifies the "feeling" expressed in that sentence (e.g. sad, happy, depressed, etc.). Because blogs are structured in largely standard ways, the age, gender, and geographical location of the author can often be extracted and saved along with the sentence, as can the local weather conditions at the time the sentence was written. All of this information is saved. We Feel Fine only collects and displays data that was already posted publicly on the World Wide Web.

It provides users with the up-to-the-moment feelings of the world. Users can search for the emotional status of a particular group of people via the "filters".

Figure 2.8 shows 6 screenshots of We Feel Fine. In Figure 2.8(1), each bubble represents a sentence or a picture that has recently been posted, and users can click on a bubble to reveal the "I feel" content. Figure 2.8(2) shows a user interface that helps the user filter feelings based on feeling keywords, blogger's gender, age, location, local weather and year. Figure 2.8(3) and Figure 2.8(4) show different forms of presentations of the top feeling key words within the last few hours. Figure 2.8(5) and Figure 2.8(6) show that feelings can be mapped according to gender and location.

2.3 Approaches towards Data-visualization Library Design

To develop exploratory visualizations, developers need to rely on data-visualization libraries, which reduce repetitive coding and promote modularized tool design. Related works on visualization library design include design patterns for reusable object-oriented software [64], and data-visualization taxonomies [154, 92, 31]. However, to the best of our knowledge, there has been no study on how to design data-visualization libraries. In this thesis, we fill this gap by making connections between data-visualization taxonomies, software design patterns and the current generation of libraries for online data-visualization developement. We discuss more detailed related work in Chapter 9.

2.4 Conclusion

In this chapter, we differentiated between three categories of related work towards online privacy, and focused on the related work for PFA tools. We identified the shortcomings of current PFA tools and motivated our developing a PFA tool that helps OSN users both explore their egocentric networks and control information flow. We then looked into the related work on visualizations tools for data analytics and exploration for aggregate data transparency, and motivated our directions in developing relevant tools. Finally, we motivated our research effort in the field of data-visualization library design.

Part I

Online Social Privacy

Chapter 3

Addressing Context Collision

Many people are on OSNs nowadays. A user's network may contain hundreds to thousands of "friends". Managing these online friends can be particularly difficult due to blurred boundaries between contexts of "conversations". This phenomenon is called Context Collision. It becomes increasingly difficult for users to act (post, chat, share, like, etc.) the way they intend to, causing undesired consequences. One of the major negative consequences of Context Collision is unwanted exposure of private information. It concerns people's social privacy on a micro-level where people conduct their daily online interactions. While the concept of Context Collision is straightforward to be understood, the concept of context itself is ambiguous. Indeed, all aspects of life can be the potential context in which a person interacts with others online. However, we argue that the most essential aspect of a so-called context is the "audience", i.e. the people whom a person talks to or shares information with, and the initial categorization of the "audience" is the first step towards context recognition and construction. In this chapter, we address Context Collision by developing an interactive friend-grouping tool named FreeBu#1 for Facebook users.

3.1 What is Context Collision?

An OSN today can hold hundreds of millions of users. Facebook (`www.facebook.com`) has exceeded its "one billion users" mark [192]. Behavioural and sometimes very personal information of OSN users is uploaded and shared online daily, in large quantities and tremendous detail. While the availability of these data enables us to understand more about our societies, it also challenges us in

effectively and efficiently processing large amounts of information, and managing our online personal content.

Context Collision or *Context Collapse* [29, 122, 140], is a widely discussed phenomenon from the world of Online Social Networks (OSNs). Boyd [29] describes it as the lack of spatial, social, and temporal boundaries (in OSNs), making it difficult (for OSN users) to maintain distinct social contexts. More specifically, the requirement to present a verifiable, singular identity makes it impossible to differ self-presentation strategies, creating tension as diverse groups of people flock to social network sites [29, 122]. This phenomenon is exacerbated when more people become users of OSNs such as Facebook (`www.facebook.com`) and Google+ (`www.plus.google.com`). According to Facebook[1], there are 1.49 billion monthly active users as of June 30, 2015.

"Context" is a multifaceted concept [48, 80]. We differentiate two characteristics of a given context: first, a context contains a group of people, and a particular role is expected from the person when he is within this context, to which we refer as the *role-playing* characteristic [147]; second, the people within the same context "are closely related to each other, in such a way that one would expect information about the user's interactions with one of them to become known to the others" [42], to which we refer as the *information-enclosing* characteristic. Therefore, the information exchange between two people in the same context is usually more private than that in different contexts. Say a person has two contexts, one is at a company where he is an employee, the other is at home, where he is a husband. He plays two different roles within two different groups of people. The conversations between him and his spouse are private to his colleagues at work and the conversations between the person and his colleagues are also usually not expected to be heard by his spouse. *Context Collision* refers to the phenomenon that the boundaries among people are blurred, the contexts in which they reside become mixed.

In an offline environment it is not difficult to distinguish between different contexts, because most of the time we know what topics we can talk about, or how we should behave toward others within a specific group of people. However, in an online environment, due to the lack of a contextualization mechanism in OSNs, such discernment is weakened. *Context Collision* makes it difficult for a user to control the flow of his/her personal information in OSNs. Following the theoretical framework of Goffman [74], De Wolf and Pierson point out that there is a lack of alternation between the front and back stages in OSNs. The front stage is where a person puts on "shows", a "polished" or "acted" self, to some extent, is presented to the "audience". The back stage is where a person communicates more privately, within a smaller range of "audience" than in the

[1] `newsroom.fb.com/company-info/`

front stage. The alternation between the two stages is necessary in constructing a personal identity. But *Context Collision* weakens user's ability to "act on stage" [184].

According to Lampe [110], the purpose of people using Facebook is primarily for maintaining their previous, offline relationships. As the PewInternet report [2] shows: an individual has met 89% of his/her Facebook friends more than once offline. Research has indicated that, by publishing (personal) information, OSN users are engaged in Impression Management [169, 76, 191] and building Social Capital [49]. Impression Management refers to the process in which people attempt to influence the perceptions of other people about a person. Typically, this person is the user herself. Social Capital refers to the resources accumulated through the relationships among people [38].

With so many complex human activities conducted in OSNs, the default contact management options offered by OSNs are poor (we examine them in more detail in Section 3.2). Besides the default options, a user often follows two options. One is to simply address a specific group of friends while others also see the published information. Alternatively, to avoid unpleasant privacy breaches, the user applies the lowest denominator strategy [95] – to only post the information that is suitable for all of his/her friends. In the former, privacy suffers; in the latter, the user's choices to realize different identities are constrained. To address *Context Collision*, we built an FA tool that helps Facebook users manage their contacts (i.e. friends).

3.2 Related Work

The increasingly large amount of data produced by our online social networking activities has made it difficult for us to manage our personal information flow. Sharing certain information with the wrong people can cause awkwardness, embarrassment or even severe damage on the user. Therefore a tool is needed to inform OSN users and facilitate their privacy decision-making. More specifically, the user should be able to effectively determine which piece of his personal information is visible to which friend(s). But it would be a daunting task if the user goes through each individual online friend that he has one by one, and considers that friend's unique constellations of attributes and proclivities in order to make such a decision. In reality, informed by the research in social cognition [119], we know that people "prefer to construe others on the basis of the social categories to which they belong, categories for which a wealth of related material is believed to reside in long-term memory". Because of the

[2] www.pewinternet.org/2011/06/16/social-networking-sites-and-our-lives/

limitations in human cognition and the challenges presented by a vast stimulus world (in our case – the online social networking environment, intertwined with the offline social life), a person naturally employs categorical thinking in order to simplify and structure the people he befriends [4, 119].

Other sociological studies [122, 140] also suggested, in order to manage personal information flow, it is important for users to categorize their online friends into groups, categories, circles, lists or communities[3], so that the user can post towards a clearly specified audience. We are interested in the tools that can help OSN users gain insight into their own social networks, explore them to reveal hidden patterns and control the flow of personal data shared with online friends. By "post", we mean the user's action of uploading or sharing digital information in OSN. We will also be using the term egocentric OSN or ego-network for short, to refer to a sub-network in an OSN, with the nodes representing people and the (directed or undirected) edges representing certain relationships among them. The network is centered on one user (as the ego), whose friends (as the alters) are directly linked to this user via edges. Edges usually also form among the friends.

A first step towards online context management is to help the user distinguish groups of friends in his/her egocentric network. Facebook, Twitter and Google+ have developed grouping features. Users can create friend lists or circles to distinguish their friends. We identify three limitations of the current grouping approaches in these major OSNs:

- *Lack of automated process*: Users need to manually construct friend groups such as Facebook lists or Google+ circles, which takes time. The only exception is that Facebook has offered users the automatic grouping function "smart list". Four types of attributes are taken into account as the grouping criteria and four corresponding smart lists are generated, namely work, school, family and city. De Wolf and Pierson [46] found that sometimes users think the smart lists are too large, not correct or not relevant, and it appears that people have different criteria in delineating their contexts. This restricted, attribute-based grouping is also unable to recognize different names of one institution. For example, people from the same school may fill in the name of their school differently, some use abbreviations, some use full name, etc. This could result in more than one smart list generated about the same school. Smart list is also limited

[3]We use these words interchangeably throughout the thesis. The words "group" and "category" are used more generically, "list" is often used in the context of Facebook and Twitter (`www.twitter.com`). We use "circle" more often in the context of Google+ (`www.plus.google.com`) and visualization. The word "community" is usually used in the context of community detection algorithms.

by the sparsity of the data, i.e. people often do not fill in the information about work, school, family or city.

- *Lack of hierarchy, i.e. one layer of grouping*: The user may want to subdivide a group to make a more fine-grained distinction.

- *Lack of visual presentation*: Facebook adopts a traditional webpage format. The user needs to click on a certain list to be directed to another page to check the group list members. Google+ has a more advanced interface with which the user can drag and drop people into different circles. The name and the number of people are displayed on top of a circle. When the mouse hovers over a circle, the photos of the people in this circle are displayed. However, more advanced visualizations revealing the user's egocentric network data structures are not provided.

Other tools or proposed solutions that enable users to gain insight into their ego-networks and/or construct friend groups include PViz, Privacy Wizard and Personal Analytics for Facebook as detailed in Subsection 2.1.1. Furthermore, NodeXL [156] is a general-purpose plugin that allows users to draw graphs by using a Microsoft Excel template. It implements various graph clustering algorithms, including modularity-based ones. It supports social network analysis: users can visualize their Facebook and Twitter graph data via an importer interface. The tool uses the Group-In-a-Box (GIB) feature [141] to help users delineate the clustering structure of the imported graphs. More specifically, the visual clusters are firstly formed in a graph layout. They are further constrained by being placed inside boxes whose sizes depend on the respective numbers of nodes. These boxes are then arranged by the squarified treemap algorithm [32]. The GIB layout is also used for multivariable grouping of the nodes based on their attributes. The GIB feature in NodeXL currently does not support hierarchical graph clustering and exploration. A hierarchy is difficult to visualize and interact with, because the semantics from different layers may compromise the readability of a set of visual clusters, especially when the leaf nodes are of main interest (e.g. the user' friends).

InMaps[4] visualizes the user's network on LinkedIn (`www.linkedin.com`) with rather similar force-directed layout and modularity-based communities as well. Through InMaps, a LinkedIn user can zoom and pan to explore the map. The name labels of the friends are simultaneously brought to display upon zooming-in. We can also see that the nodes and labels are mapped with care to avoid overlapping. Some other general-purpose network-analysis tools are potentially useful for OSN users as well, such as Gephi [13], Cytoscape [149] and Tulip [6].

[4] `www.linkedin.com`

We find that the graph-based community detection approach is popular among PViz, Personal Analytics for Facebook, NodeXL, InMaps and all the general-purpose graph-visualization tools. But in the published materials that describe the corresponding implementations, none directly motivates the reason why such approach is adopted. Bacon and Dewan [8] mention that the friends' detailed information is largely not filled out by the users and that smart list cannot distinguish synonyms. It also appears that few have provided an informative graphical user interface to the user with the meta-information about the detected communities. Meta-information is the information about the properties or characteristics of the formed communities. A user can be informed of such information by various visual cues, including textual labels, which only PViz has applied to its user interface. However, full automation of friend community recommendation is not advisable since a user may group his/her friends based on a variety of reasons. It is important to leave room for manual and interactive adjustment by the user.

3.3 Research Questions

While there are abundant options for OSN users to explore their social networks or create friend groups, none has motivated its implementation from the user perspective. In order to provide sensible grouping recommendations to a user, we need to understand the criteria users apply to grouping people. We first ask:

RQ1: How does a person categorize the people he/she knows?

Moreover, because existing tools are either without facilities (mainly visualizations) that enable OSN-data-exploration, or with static visualizations that cannot be used to dynamically create friend groups, we ask:

RQ2: How to we develop a semi-automatic friend-grouping visualization tool for OSN users to categorize their friends? And

RQ3: What are the users' perceived values towards our tool?

Answers to these questions will be given in Section 3.4 (RQ1), Section 3.5 (RQ2) and Section 3.6 (RQ3) respectively.

3.4 Users' Grouping Approaches (RQ1)

Social networking sites are structured as personal networks: one OSN account corresponds to one individual, who, as the center of the network, interacts with

his/her friends online. Hence, we base our user-study approach on an egocentric perspective.

In the user study, we asked a participant to group the people he/she knows personally. We denote each person with E, as in "Ego", and the people E knows personally are denoted as $F(E)$. Note that $F(E)$ may include both online and offline contacts of E, since E's OSN contacts do not cover the whole set of E's contacts in his/her life. By including offline contacts, we encourage E to think independently of OSNs, to reflect upon what is essential for him/her to create boundaries/ contexts via such grouping, in the hope of discovering a grouping structure that is not distorted due to the limitations of online platforms.

However, one common question asked before any of the groupings to be performed is: "Why do we want to group the people we know?", or "what is the purpose of the grouping?". It may appear to a user that, without any particular purpose defined, it is not sensible (or even possible) for him/her to create such a grouping. Some argue that different purposes yield very different grouping structures. While forming friend groups is necessary to protect the user's privacy and help the user present herself appropriately online, the purposes of the formed groups can be different. For example, if a user wants to form a book club, he might group the people he knows into two categories, solely based on a book, one in which people like the book and the other not. The rest of the attributes of the people simply do not matter to the user in this purpose of grouping. We call this kind of grouping the specific-purpose grouping, which only arises from a specific occasion and only considers one or a few specific attribute(s).

We argue that there is another type of grouping, which is created when we don't bear any particular purpose in mind. Instead, we take a holistic view towards the people we know and make general divisions among them. We call this general-purpose grouping. Such grouping can provide an immediate impression of $F(E)$ that is considered clear and sensible, so as to well illustrate and summarize the connections or relationships of E. Also, because of its generality, E can almost always construct other specific-purpose groupings by starting from the general-purpose one.

3.4.1 Participants and Method

We asked 15 participants to group the people they know in their lives. Out of the fifteen, six are PhD students in Computer Science, three are employees from different companies, the other six are university Bachelor students. 40% of the participants are females. The ages range from twenty-two to thirty-one. Note that our selection of the users is mainly limited in terms of age. Nonetheless,

given the fact that young people are driving the usage of Facebook[5], we believe that this study holds value in aiding group-building in OSNs.

Each E was asked to draw a grouping structure of $F(E)$, in a star-tree form. If E was not familiar with the concept of star tree, an example was given, as in Figure 3.1. The participants were asked to construct the drawing on a computer with a drawing tool of their choice or on paper. We gave the following guidelines:

- Group the people in your life, who you know personally and is alive.
- You are supposed to be in the middle as the "self" in the example from which all the curves (i.e. branches) sprout.
- Hierarchy is allowed, i.e. a branch can split into more branches, each branch is considered as a group, the subbranch as the subgroup, and so on.
- The label on each branch characterizes or summarizes the people within that branch; if a branch contains only one person, a label may not be required.
- The same person can appear in different (sub) branches.
- Proceed with the grouping until adding more people would not require new (sub) branch(es).
- Labeling the tips of the branches with names is encouraged, as illustrated in Figure 3.1, but not mandatory.

3.4.2 Results and Interpretation

We analyzed the fifteen general-purpose groupings from the participants by categorizing their labels and counting the frequencies of the labels. We categorized the labels used in the groupings into ten categories: interest/hobby (hobby), education (edu.), work, social community (comm.), language/nationalities (lang.), location (loc.), time, age, family and connection (conn.).

Interests/ Hobbies labels indicate a group of people performing recurrent activities based on common interests, including keywords such as "skiing", "rugby", "concert group", "travel buddies", "pingpong club", etc. Education-related labels include school, university and major names, degree titles, or

[5] www.pewinternet.org/2015/01/09/demographics-of-key-social-networking-platforms-2/

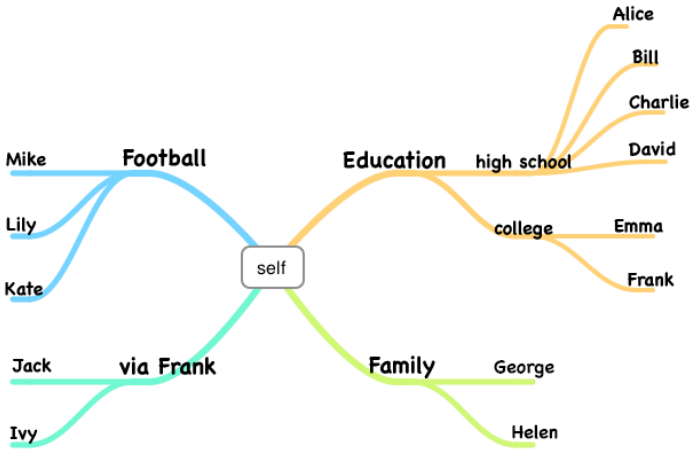

Figure 3.1: A grouping-tree example.

keywords such as “classmates”, “fellow students”, “professors”, etc. Work-related labels mainly cover company names, project names, and keywords such as “work”, “leaders”, “colleagues”, etc. Social-community-related labels indicate that people are grouped based on organizations that are different from schools, universities and corporations, e.g. a youth-movement organization. The labels of Language/Nationalities are used to distinguish groups of people with different nationalities, languages or ethnic backgrounds. Location-related labels categorize people based on locations, the keywords are city names. Time-related labels specifically indicate a period of time, such as “current”, “old”, “childhood”. Age-related labels include “elderly”, “peers”, “senior”, “junior”, etc. Family-related labels cover keywords such as “relatives”, “(not) related-by-blood”, “parents”, etc. Connection-related labels indicate the strengths or types of connections that E has with other people. The keywords include “close”, “best”, “(dis)like”, “acquainted”, “not interested”, etc., or secondary connections that express E knowing a group of people via a specific person X, who acts as a bridge and is considered an important grouping indicator, the keywords of secondary connections include: “X’s friends”, “X’s connections”, “via X”, etc.

To measure the importance of each category of the labels, we count the frequency of the labels in each category. The initial counting result is shown in Table 3.1. The columns are the categories of the labels, the rows are Es. Note that some people perform the grouping in a more detailed way than others, i.e. with more labels. For example, if E happens to have a big family with various relatives,

Table 3.1: Label counts of different categories for each E

	hobby	edu.	work	comm.	lang.	loc.	time	age	family	conn.
E1	1	8	1	12	0	0	2	2	1	5
E2	0	4	1	0	2	0	0	0	4	3
E3	0	2	1	0	0	0	0	0	9	7
E4	3	6	1	1	0	0	0	0	1	44
E5	10	6	7	0	4	4	0	4	5	6
E6	2	3	4	0	1	1	0	0	1	4
E7	3	11	7	1	0	0	0	0	4	5
E8	0	0	1	0	0	0	0	0	3	2
E9	0	8	0	0	0	0	0	4	3	15
E10	0	6	0	0	0	0	0	4	3	8
E11	0	5	0	0	0	0	0	0	3	6
E12	0	4	1	0	0	1	0	1	4	12
E13	0	14	0	0	0	0	0	0	3	1
E14	2	0	0	0	0	0	0	0	1	15
E15	1	6	3	0	2	0	0	0	4	7

this situation may force him/her to put more family-related labels into the star tree, which gives us the impression that he/she emphasizes the importance of the family-related labels, while he/she actually considers the other categories of labels just as important.

To compare all the counts on the same scale, each E's counts are divided by his/her total number of counts respectively, deriving percentages. Then, we calculate the average percentage for each category of labels, the result is shown in Figure 3.2. On average, each E uses 36% connection-strength labels, 24% education labels, 18% family labels, 8% work labels. Note that some labels may have implications for others. For example, the people marked with work labels tend to be older than those marked with education labels. Nonetheless, we consider only the label types that are explicitly determined by the participants, not the correlations between different label types.

We see that Es perform the groupings primarily based on their connections with $F(E)$, they consider the types and strengths of these connections important. For example, from Es' groupings, we observed that close friends and acquaintances were often prominently distinguished, or, a group of people who E got to know through a friend was often emphasized, preceding other criteria. Also, the connection-related criterion and others often work in a combined fashion. We observed that some Es first grouped $F(E)$ according to schools, then within

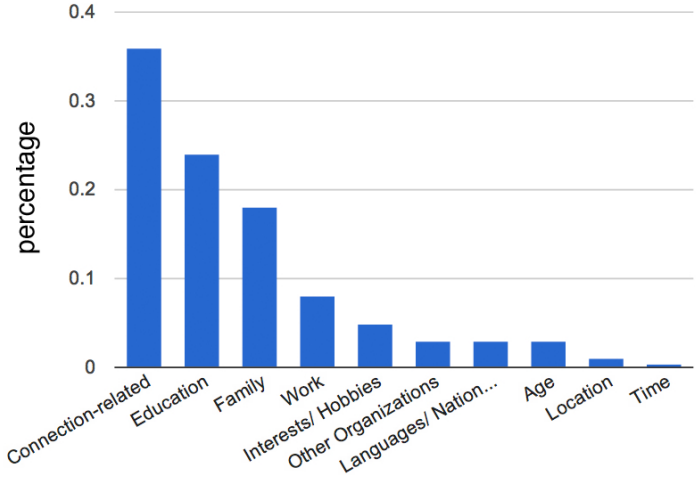

Figure 3.2: The ranking of the label categories

each school, people were divided based on connections, e.g. the closed ones and acquainted ones.

3.4.3 Implications for Tool Design

From this study, we can see that a person's connections towards his/her friends are the primary criteria for constructing friend groups. However, it is not straightforward to derive information on the types or strengths of connection solely based on social network data, due to its ambiguous and subjective nature and the sparsity of the network data.

The attributes on school, work and hobby are often available in users' profiles, but when automatically constructing friend groups only based on these attributes, we lose the opportunity to provide users the groupings on the other criteria, especially the connection-related ones. However, we can see that criteria such as education, family and work are closely related to connection-based ones, because people are more likely to form connections in the same family, school or workplace than otherwise, and a person's opinions or feelings of closeness towards his/her friends are likely to be associated with attributes such as education, work and family. Furthermore, given the popularity of the groupings based on secondary connections, we can see that in a grouping, E considers not only his/her connections with $F(E)$, but also the connections among $F(E)$. Also, an E's friend graph can be more informative than profile attributes. While the

attributes convey simple facts – education, family, work, hobbies, languages, etc. – about people, the social network graph may imply more complex relationships that are formed and shaped through a history of interpersonal interactions, and cannot be readily expressed by the attributes.

Therefore, a work-around is to use graph-based community detection algorithms to produce grouping recommendations. We make the assumption that it is more likely that the friends in the same group have more mutual linkage than on average. For example, a person's family members tend to be linked to one another, just as school mates tend to be mutually linked. Based on this assumption, our grouping tool used Newman's concept of modularity [134] and Blondel's algorithm [26] to find such communities given an egocentric social network graph.

Finally, because the number of participants is small, and the demographic structure of the sampled people in the user study is limited, mainly in terms of their age range, but also educational and cultural backgrounds, this study serves as a useful starting point for us to inform the design choices of our grouping tool. More specifically, we select a user's Facebook data according to the grouping criteria for label derivation and adopt a graph-based community detection algorithm.

3.5 FreeBu#1 (RQ2)

To address RQ2, we developed a semi-automatic friend-grouping visualization tool called FreeBu#1 (as there has been a series of new versions afterwards, see Appendix C), which is short for Friend tree Bubbles.[6] This section describes the method that we use to recommend an initial grouping of the user's OSN friends, while enabling the user to explore and adapt this grouping according to his/her needs, and eventually publish the results onto his/her OSN account. Because of the prevalence of Facebook among today's OSNs, we have implemented this tool based on Facebook data. The method is however applicable for any other OSN that provides data access to its social graph and profile data.

[6]Note that because of Facebook API change, as documented in `https://developers.facebook.com/docs/apps/changelog`, the three versions of FreeBu: FreeBu#1 (Chapter 3), FreeBu#2 (Chapter 4) and FreeBu#3 (Chapter 6) are no longer functional. To avoid applications like this being dependent on a particular OSN's API, we redesigned FreeBu so that it now functions as generic tool for graph data visualization and exploration, as detailed in Chapter 10.

3.5.1 Data

We base our PFA tool on the data retrieved via the Facebook graph API[7] with the user's access token. We aid the user to group his/her Facebook friends, by firstly recommending an initial grouping structure, assigning appropriate labels to the groups and then letting him/her further adjust the grouping (see the following sections on Computational Model and User Interface). The grouping is constructed based on the user's friend graph, in which each node is a friend of the user's, and if two friends are also friends to each other, they are linked. Note that such links are unweighted. We generate labels for the groups with collected attribute-based data.

Education-related data includes a list of schools, where each school has its name and type, e.g. high school or graduate school, with possibly more information such as the year and the concentrations if the user has filled this in. Work-related data includes a list of work-objects, each object contains the name and the location of the employer, the position of the user, and the starting and ending time of the job. Language-related data includes a list of names of the languages that the user speaks. For hobby-related data, we collect the "likes" of a user, which may contain anything, from sports to TV shows, from a public figure to a book, etc.

It is important to let the user form sensible groups on his/her own, by providing options of available OSN data on a meta-level. For example, what data attributes are available on OSN, and how are people distributed over these attributes. Eventually, we let the user decide what data is most relevant and what grouping structure is closest to what he/she has in mind.

3.5.2 Choosing A Community Detection Method

Based on our user study results in Section 3.4.2 and discussion in Section 3.4.3, we adopt a graph-based community detection algorithm – more specifically, the Louvain method [26] – to extract communities from the user's friend graph. It is a heuristic method that is based on modularity optimization. The method was shown [26] to be efficient and produce communities with good quality. A community is characterized by modularity. Modularity measures the density of links inside communities as compared to links between communities [134]. An area with more mutually connected friends is more likely to be identified as a community. The Louvain method outputs flat communities.

[7] `https://developers.facebook.com/docs/authentication/`

Note that we detail the comparison between the modularity-based community detection method and the method that takes both graph and node-attributes into account in Chapter 5. This comparison further showed that the modularity-based method is indeed suitable for detecting communities in ego-networks.

3.5.3 Label Derivation

To support the exploration of the visualization, and help the user identify the characteristics of different groups, it is critical to derive informative labels for communities. The label of a group should highlight the attributes of the people in it. We adopt the *F-measure* to determine the labels for the communities. *F-measure* is a standard measure combining precision and recall (Equation 3.1). As the labeling experiments in [123] indicate, *F-measure* comes out as one of the best label-selection measures for communities detected with the user's Facebook data – the labels with high *F-measure* scores are generally considered suitable by the users.

$$F\text{-}measure = 2\frac{Precision(C,A) * Recall(C,A)}{Precision(C,A) + Recall(C,A)} \tag{3.1}$$

with

$$Precision(C,A) = \frac{|C \cap A|}{|A|} \tag{3.2}$$

$$Recall(C,A) = \frac{|C \cap A|}{|C|} \tag{3.3}$$

C denotes a set of people within the same community c, A denotes a set of people with the same attribute-value a, e.g. a certain name of a university. $Precision(C,A)$ measures the proportion of people with the attribute-value a in the community c to the whole population with attribute-value a. $Recall(C,A)$ measures the proportion of people with the attribute-value a in the community c to the whole population of the community c.

For each community, a list of labels is generated based on all the data attributes described in Section 3.5.1, and then sorted according to every label's *F-measure* score. The user can determine the number of labels appearing on the communities. The labels with higher *F-measure* scores are selected first.

3.5.4 Visualization Interface

We adopt the star-tree form to represent the grouping structure. As shown in Figure 3.3, the nodes of the tree are represented by circles, each pair of parent-child nodes connected by straight lines. The root of the tree (the blue circle in the middle) is the user herself, the red circles represent different communities detected by the algorithm, the leaves (the green circles surrounding the red ones) represent the user's friends on Facebook. We scale the sizes of community circles based on the number of people within each community, a larger size corresponds to more people.

The labels are shown on top of the community circles, if a community contains more than one person. The user can click on one bubble – a community or a person – to zoom in to concentrate on a particular part of the tree. The labels are typically school names, school years and work places. The number in front of the labels indicate the number of people in the corresponding circles. The user can adjust the number of labels shown by sliding the threshold bar. The user can turn the labels for the red and green bubbles on or off via the "rlabels" and "glabels" button. The switch for the tip (an info-box) appearing on top of a bubble is the "speech" button. The user can also press the "shuffle" button to rearrange the layout of the bubbles.

Initially, we provide the user with a one-layer grouping. The user can modify it by adding or removing (sub) groups via the "add" and "delete" buttons, so that the user is able to construct his/her grouping hierarchically, as shown in Figure 3.4. The user can specify the name for the newly added group in the text box below the "add" button (Figure 3.3). The user can also change the members of the groups by "dragging and dropping" friend nodes from one red circle to another, as shown in Figure 3.5. Once the user finishes modifying the groups, he can publish the grouping to his/her Facebook account as "Facebook friend lists" by pressing the "publish" button. However, note that Facebook friend lists are flat groupings. Thus only a one-layer grouping is composed and published.

3.6 Perceived Values of FreeBu#1 (RQ3)

In this section, we describe the user study that investigates the perceived values of FreeBu#1. The content of this section is mainly from the joint work [45] with researchers from SMIT, VUB[8].

[8] `smit.vub.ac.be/`

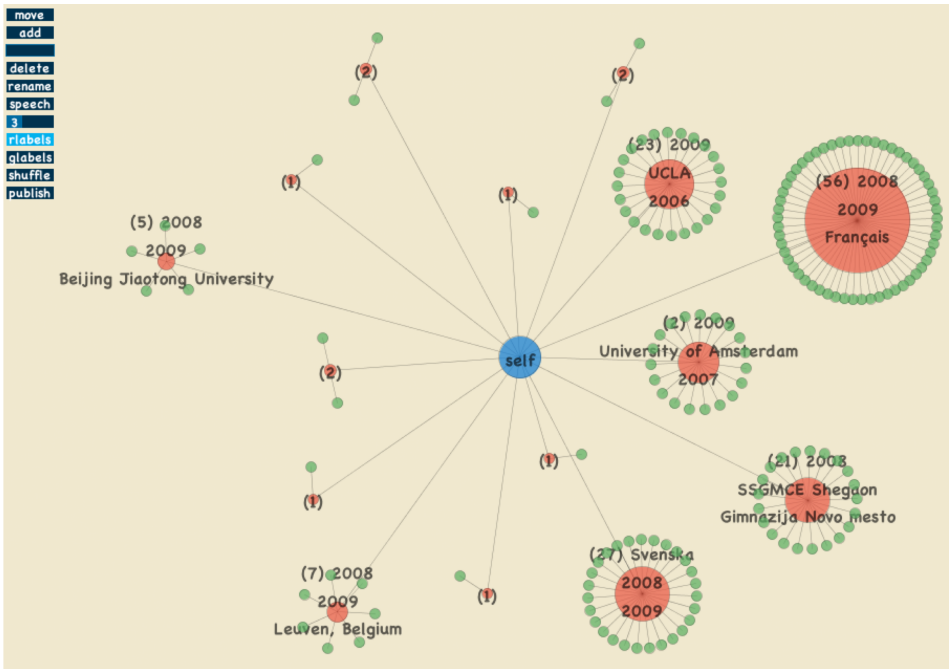

Figure 3.3: The overview of the visualization interface

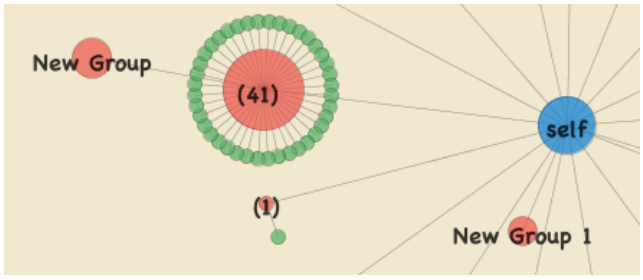

Figure 3.4: The user can add new groups at different levels of the star-tree, “New group” is added at level two, attached to the level-one circle labeled with “41”, “New group 1” is added at level one, directly attached to the “self” circle. The user can also edit the labels of the circles.

3.6.1 Participants and Method

We selected adolescents and young adults as the population of our joint study. People in this age group typically go to work or go to college, which increases

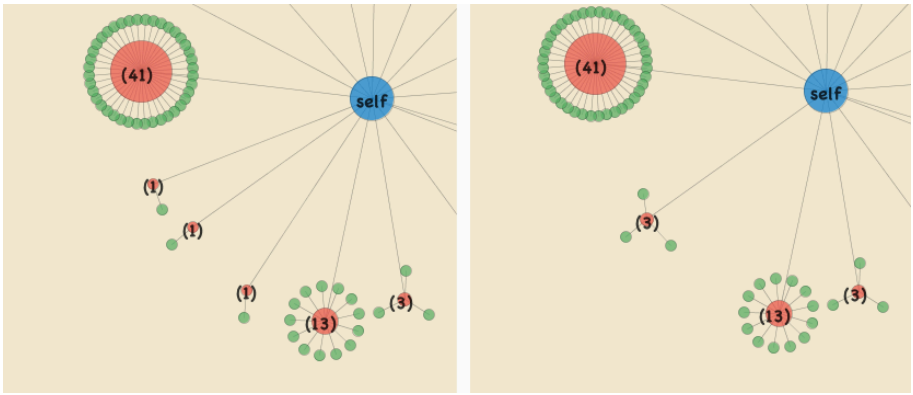

Figure 3.5: On the left, three individuals are initially assigned to three different groups. On the right, the user move the three individuals into one group.

the presence of multiple types of audiences. The participants were recruited by reaching out to different intermediaries who are in close contact with adolescents and young adults. More specifically, the heads of different youth organizations[9] were contacted to spread the call for participating in the user study of FreeBu#1. The participants were compensated with a cinema ticket. Twelve people (7 male/5 female; 17–23 years old) participated in the first study. They were active Facebook users and each had more than 300 Facebook friends.

FreeBu#1 was installed on the computer of the participants, who explored the tool whilst being asked questions from the semi-structured topic guide (Appendix A.1). We deliberately did not provide any task-based instructions but let participants interact with the tool in an open way in order to understand their perceptions. The interviews were audio-recorded and transcribed afterwards.[10]

In order for FreeBu#1 to function, a participant's personal data was retrieved via the Facebook Graph API, using his/her access tokens. The participants were explicitly informed on the installation procedure and asked for their consent. Under the guidance of the researcher, participants first logged into their Facebook accounts. Then, they retrieved their access token through the Facebook graph API explorer[11], with which they generated access tokens with required permissions, including user's and friends' profiles and graph data. The participant then copy-pasted the access token into FreeBu#1 for it to download the data onto the participant's personal computer and produce the visualization.

[9] such as KSJ-KSA-VKSJ Nationaal (`ksj.be/`)

[10] The interviews were conducted, and the corresponding materials as in Appendix A.1 and A.2 were produced by Ralf De Wolf.

[11] `https://developers.facebook.com/tools/explorer`

During the interviews, we used a grounded theory approach when coding the transcriptions [40]. The first phase of coding was data-driven: we maintained a close connection between the codes and data and coded the one word that appeared significant. In a second phase we categorized these codes with labels such as "functionality", "appearance" and "usability". In a third and last phase, we further organized and merged the codes and outlined different dimensions and properties where necessary. The coding process is summarized in Appendix A.2.

3.6.2 Results

We identified four affordances as perceived by the participants: (1) friend-removal, (2) friend-grouping, (3) overview, and (4) reflection.

Friend-removal: During the interviews many participants indicated that it was difficult to remove friends using the settings provided by Facebook. Several indicated that FreeBu#1 would make it easier to identify "outlier friends", who often did not have any affiliation with the participant, or were not familiar to them at all. For example, one of the participant said: "I think I know this guy through playing an online game, but I do not really know him." Another participant noticed that he had befriended two persons with the same name, and that only one of them was familiar to him. Like the first participant, he would also unfriend this person from his/her Facebook account.

Friend-grouping: Participants indicated that it would be easy to share certain information with a specified audience. For example, one participant said "These people are all involved in my youth movement. This would make it easy to post something just for them."

Overview: Most participants also identified that it was valuable and enjoyable that FreeBu#1's visualization provided an overview of one's audiences. For example, when asked whether he/she would use the tool to limit information access towards certain audiences, the participant replied: "no, it is just fun to see your friends like this." This is also in accordance with the argument that people are inherently curious about their online social data [28].

Reflection: We noticed that the participants often reflect on the visualized groupings. They try to infer the meaning behind groups, and they also point out the aspects that do not "make sense" to them. For example, one participant said: "This one is spot on. I know all of these people via my girlfriend. This one is her cousin. My girlfriend herself, however, is not in this category". Another participant mentioned: "This group is not correct, different people are just thrown together. Maybe they are all residents of Ghent."

3.7 Discussion

In this section, we discuss the limitations and future work of FreeBu#1.

3.7.1 On the Data

We use friend graph data, in which a connection between two people is formed when the two are friends of each other. However, such friend graph data is not ideal to describe users' relationships, as we do not know the type and the strength of a connection in a given friend graph. A connection may be formed because of many reasons. For example, a pair of individuals have chatted pleasantly in an offline meeting and decide to become friends on Facebook, or two oldest and closest offline friends one day add each other on Facebook, or they have never met each other offline, but both are actively in an online forum, and then became Facebook friends.

As the friend graph lacks detailed connection information, i.e. type and strength, it may become an issue. For instance, someone A in the user's friend graph is categorized into a community because A has more links with the people in that community, meanwhile A also connects to B who is in another community. The user considers the linkage between A and B is much stronger than the rest of A's links to others, and A should be put into B's community, or simply A should be in both communities simultaneously. The friend graph does not contain such knowledge. Further investigations may focus on the measurement of the strengths of connections and the involvement of the user's input, making the friend graph more informative.

Another limitation of the friend-graph data is that the user does not group his/her friends purely based on connections, but sometimes on attributes as well. However, we do not know under what circumstances the user chooses attributes over connections to group friends. In the next step, we can offer different versions of grouping recommendations to the user, including purely connection-based, purely attribute-based grouping, and a mixture of both, so that the user is reminded with more alternatives and acquires more insight into the grouping structure of his/her friends. Also, in this process, the user's choices could be stored and better groupings could be provided based on the user's preference.

One other limitation of the data holds in OSNs in general – an online OSN is not a perfect replica of the offline world. Usually, critical data in an OSN is missing, or the OSN is not synchronized or updated with the offline counterpart. Some important offline activities or events may not be shown on Facebook, a

user may not fill in a certain hobby on Facebook but indeed practice this hobby often and have a special group of friends.

3.7.2 On Community Detection and Visualization

The graph-based community detection algorithm derives communities by optimizing modularity. However, modularity optimization is computationally hard [30]. Sometimes it is necessary to adopt approximation algorithms to deal with large graphs, as does the Louvain method. Also, the Louvain method outputs a flat grouping structure for each given graph, while a hierarchical grouping structure may enhance the user's comprehension of his/her contexts.

Also, there is a known resolution limit [61] in modularity-based groupings. This was also reflected in the interviews with the participants. Most of the time, the small groups were considered "correct", while the larger groups were perceived as a merging of different groups. For example, one interviewee said: "I could certainly make further categorizations. This guy used to be a member of our movement. This one is still a member. This one I got to know on Expies (a camp for youth movements). So I would make a differentiation between all of these."

We also find that randomly spreading the branches around the central node in the star-tree visualization is not ideal. It often produces overlapping nodes.

Moreover, as suggested in [34], there can be friends who bridge different groups, and algorithms and visualizations that emphasize on discovering bridging/overlapping structures are potentially useful.

Finally, as discussed in Section 3.4.3, the connection-based criteria for people grouping is favored by the participants. But it is not always the case – there are times that people do purely attribute-based grouping. To make more fine-tuned grouping recommendations, we should extend the visualization so as to directly visualize groupings based on attribute data on nodes and edges (such as the types and weights of the edges, which can be derived from the commenting or the internal chatting messages of a user in OSNs).

3.8 Conclusion

In this chapter, we introduced an interactive grouping tool FreeBu#1 using the user's Facebook data. We investigated the criteria people applied to grouping the people they knew, identified several data attributes that users frequently

adopted for labelling their groups, and the inherently connection-based grouping method by users. We developed the grouping tool based on our findings in the user study that could help users recognize and modify the grouping of people. With the automatic community detection and labelling features in the tool, the user gained an overview of his/her friends on Facebook. The user could then adjust this grouping structure by adding, removing groups, rearranging the members inside the groups, and eventually apply the grouping decision onto his/her Facebook account. The grouping tool helped the user create sensible boundaries in the OSN, which made it easier for the user to decide what to publish or receive (sometimes very sensitive or personal) information to or from which group of people. We consider the interactive grouping tool as a first step to address the privacy concern *Context Collision*.

Chapter 4

Using Friend Groups to Avoid Regretted Posts

Due to Context Collision in OSNs, users may share sensitive, often private information with the "wrong" friends, leading to many regrets. In Chapter 3, we developed an semi-automatic friend-grouping tool FreeBu#1 for Facebook users. We found that users valued this tool for its interactive visualization that can help them to not only easily identify the "friends" they want to remove, but overview and reflect on their Facebook relationships. Users also valued its friend-grouping functionality. Among the limitations of FreeBu#1 discussed in Section 3.7, there are two major limitations of the modularity-based community detection method: resolution limit and non-hierarchical communities. We also found that the star-tree visualization often generated overlapping node positions, which introduced occlusion. To tackle these problems, we redesigned the visualization and reimplemented the tool with standard Facebook login. We named the tool FreeBu#2.[1] Its key feature is to allow the user to interactively divide a group of friends into smaller groups with modularity-based community detection.

The purpose of friend-grouping is to send/receive information to/from more clearly defined friends, and ultimately avoid regrets. In this chapter we motivate and describe the design of FreeBu#2 and investigate how the modularity-based community detection method used in an interactive way in FreeBu#2 can help users better avoid regrets than traditional Facebook smart lists.

[1] For an overview of the different versions, refer to Appendix C.

4.1 Related Work

In this section, we will be reviewing the related work on the cognitive reasoning behind friend-grouping applications (which supplements the content in Section 3.2), and the related work on visualization designs for hierarchical data.

4.1.1 On Friend Grouping

Social and cognitive theories shed light on human social grouping behavior and inform computer scientists to design community detection algorithms and interactive visualizations. The social brain hypothesis (SBH) offers a framework for integrating evolutionary and social psychological perspectives on human social complexity. SBH predicts a natural community size of around 150 for modern humans (Dunbar's number [51]), and now there is considerable evidence confirming that this is the typical size of both personal social networks and key types of human community [50]. Note that 150 is the typical size of a person's active network, in which she knows how these the friends fit into her social world and they know how she fits into theirs [50]. From the literature in cognitive science, we also know that there is the cognitive capacity limit in human Short-Term-Memory (STM), which is inline with the theory of categorical thinking (Section 4.1.1). This capacity limit is averaging on seven [125], which means that people can remember seven chunks of information in STM tasks. In our case, we can consider a chunk to be a group of friends. This limit is subject to debate, later evidences showed that it was a high estimate, lower numbers were proposed, e.g. four [41]. The theories on social group size and human's cognitive capacity limit provide more incentives for interactive visualizations, which should enable users to flexibly interact with friend visual objects on different granularity-levels – from (sub-)groups of friends to individual friends.

The related tools on contact management in OSNs have been documented in Section 3.2. By large, users have to manually group friends, which tends to become unmanageable. We know one exception – Facebook Smart Lists (FSL)[2], which provide users with an automatic grouping solution. The lists are generated based on the information about the user's education, work and current city. For example, if the user indicates Leuven as his/her current city, he/she will have a list with all of his/her friends who also indicate Leuven as their current city. The user can directly determine the audience of his/her posts by choosing one of the lists, including the smart lists. Figure 4.1 gives an example for status update.

[2] `https://www.facebook.com/help/204604196335128/`

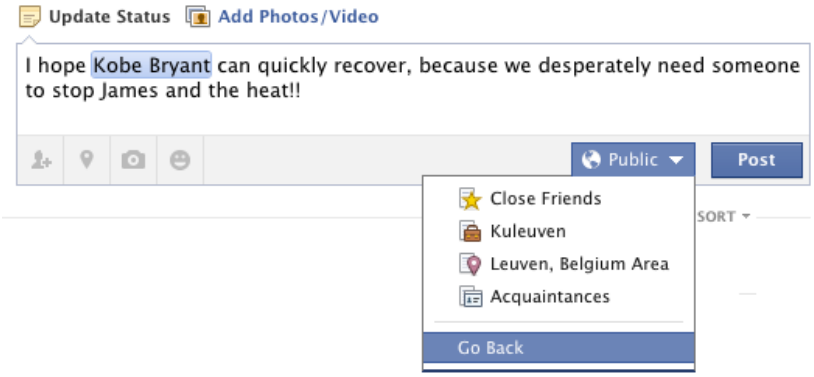

Figure 4.1: A Facebook user can conveniently limit the visibility of his/her status by choosing one of the four lists, of which *Close Friends* and *Acquaintances* are the lists that the user manually defines, and *Kuleuven* and *Leuven, Belgium Area* are the automatically generated smart lists, based on the user's work and current city.

However, as discussed in Chapter 3, FSL need the attribute information in a user's profile to construct lists, but this type of information may often be missing as users do not fill out their profile forms. More importantly, users may not need attributes such as education, work and current city as the basis for constructing friend lists, as suggested by our user study in Section 3.4. Freebu#1 adopted the modularity-based community detection method (MOD) to produce an initial set of friend groups and let its user subsequently modify the groups. Because of the following three reasons, we adopt a visualization that allows the user to interactively divide a community into sub-communities.

- MOD methods are known to have a "resolution limit" problem [61]. It is most likely that, for a community with $\sqrt{m}$ (m is the total number of edges) or less nodes, its sub-communities cannot be discovered. This implies that modularity optimization can miss the substructures of a network.
- It is well known that people organize semantic concepts hierarchically in memory [39]. The reason for this is because storing generalized information with superset nodes is more economical for humans. Hierarchy is necessary in the navigation for the retrieval of more detailed information.
- Another incentive is based on the aforementioned Categorical Thinking, as iterative grouping may be required from the user to make sense of the his/her friends if the number of friends is simply very large.

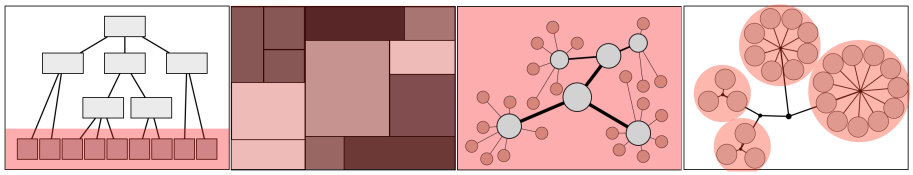

Figure 4.2: Four types of representations for visualizing a hierarchical grouping structure, the potential area that can be used to draw leaf nodes is overlaid with red color.

More specifically, the original MOD takes the user's friend graph as input and produces non-overlapping, flat communities. There is a subgraph corresponding to each detected community of nodes. MOD is then applied to each subgraph, deriving sub-communities. We adapt MOD into a hierarchical variant, abbreviated as HMOD.

4.1.2 On Visualization for Hierarchies

Various existing works have paved the way for visualizing hierarchical grouping structures. We do not intend to provide a comprehensive review in this subsection. Instead, we give a qualitative treatment to four representative types of visualizations and motivate our design choices. We refer to the two dimensional area on the computer screen where a visualization is rendered as the canvas.

- **Node-Link Diagram** The traditional Node-link Diagrams use shapes (rectangles, circles, etc.) to represent nodes and lines to represent links. The direction from the root of the tree to the leaves is either vertical or horizontal. The nodes (intermediate or the leaves) at the same level need to be aligned at the same vertical or horizontal line. Hence only one-dimensional space is utilized to visualize each level. As shown in Figure 4.2a, this space can be easily exhausted, especially at the leaf level. When the leaves are squeezed to be aligned and fit into the canvas, they easily become too small for the user to interact with, and the grouping structure is no longer clear at the leaf level. Improvements have been made using coloring and merging to reduce the number of branches and/or leaves to draw (e.g. Colored trees [148]). However, they leverage the continuous values of leaves, so that the colors correspond to different average values, giving a sense of numerical ordering. In our case, either the friends or the groups are discrete, which the user needs to differentiate to make a

visibility decision. We also note that using the color visual channel to differentiate discrete variables (e.g. Stacked Tree [23]) is problematic, as there are very limited choices for visually distinct colors [90, 78].

- **TreeMap** Grid-based (or matrix-based) visualizations utilize the canvas space more efficiently, as shown in Figure 4.2b. A typical grid-based layout is treemap [99]. It visualizes hierarchical data by nested rectangles. Many techniques have been proposed to make treemaps more structurally perceivable by humans. For example, shaded colors can bring a sense of ordering to the treemap nodes [157], gradient colors can demarcate different clusters in a treemap (cushion treemap) [174]. And the aspect ratio of the nodes can be adjusted to improve their readability (squarified treemap) [32]. Compared with node-link diagrams, treemaps are more readable for various large-graph-related tasks, but path finding is consistently in favor of node-link diagrams [73]. More importantly, the user cannot conveniently select all the friends in a (sub-)group at once to make a visibility decision in treemaps.
- **Space-Filling Tree** Given the limitations in node-link diagrams and treemaps, hybrid visualizations have been proposed. The space-filling tree [136] is a typical example, as shown in Figure 4.2c. It spreads the nodes and leaves across the whole canvas. To give a sense of structure, the sizes of the nodes decrease with ascending levels of the tree, and the child nodes are mapped in proximity with their parent. However, it is probable that, in order to optimally utilize the unoccupied space, the nodes in one branch protrude into the neighborhood of another branch, resulting in a less structural display.
- **Bubble Tree** To heighten the sense of grouping structure, Bubble Tree [79] further constrains the proximity mapping between child and parent nodes – the child nodes are aligned in a circle around their parent, as shown in Figure 4.2d. This sacrifices potential drawing area on the canvas (still more space-filling than the traditional node-link diagram), but gains the representation of a stronger grouping structure. The user can select a branch of nodes via their parent node. However, it remains difficult to compare the sizes of the branches on the same level.

4.2 Research Questions

Based on our review of the related work, we have the following research questions:

RQ1: How to design an interactive visualization that can trade-off efficient usage of space against hierarchical visual clutter?

RQ2: What are the common regrets among Facebook users?

RQ3: Which grouping strategy is more useful to Facebook users, HMOD used in an interactive way or FSL?

We address these questions in Section 4.3 (RQ1), Section 4.4 and Section 4.5 respectively.

4.3 FreeBu#2 (RQ1)

Considering the previously examined visualizations in Section 4.1.2, we realize that showing the complete structure of a tree of friends may be unnecessary, even interferential to the user. As we try to facilitate the user in determining whether a friend (represented by a leaf node) can see his/her post, drawing too many intermediate nodes on the canvas produces unnecessary "cognitive overhead" [15], because those nodes not only occupy limited canvas space, but also increase the number of objects that the user needs to process in the limited short-term memory. Therefore, we design a new form of interactive visualization that constrains the number of levels shown (namely one or two levels) and let the user's zooming actions reveal more sub-groups or less only when he needs to. It is also similar to the Bubble Tree in the way that child nodes are positioned in a circle around their parent.

The main purpose of our visualization design is to provide visual tokens for a user's friend grouping. It adds the elements of structure and engagement to an otherwise lengthy, textual reading and decision-making task. Another purpose of the visualization is to facilitate manual friend-grouping construction. The visualization and its interactive functions are detailed below.

As shown in Figure 4.3a, a node is represented by a circle, and a group of nodes is represented by the circular placement of its child nodes around one extra node, which is the parent node that represents the whole circle. With the basic visual principles in mind – that humans are very sensitive to the difference of lightness in grey colors [161], we set the background color white, the friend nodes grey, the parent nodes blue. The latter two colors are also semi-transparent to avoid the occlusion effect. We pick orange and magenta as the highlight color for each friend node and parent node respectively. The large differences (from grey) in saturation and (from blue) in hue promote visual contrast [177]. At first sight, it seems sufficient to use just one visual channel to encode grouping, i.e. the circular placement of child nodes in a group. But since the user is allowed to drag the nodes to other positions on the canvas, as described below, we add lines connecting the child nodes with the corresponding parent to emphasize

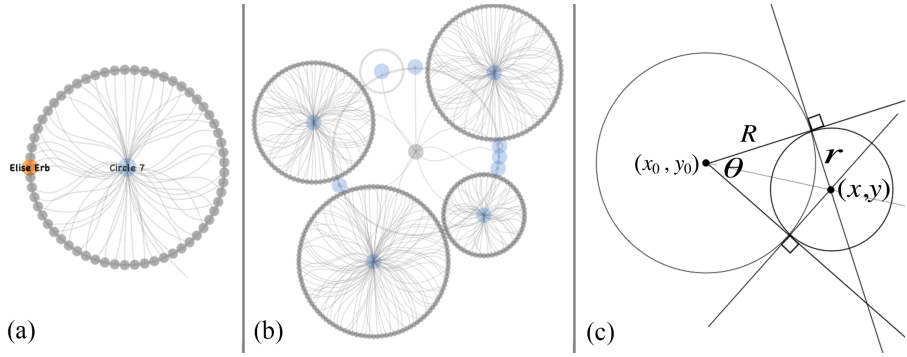

Figure 4.3: (a) A single group circle, the grey nodes are the friend nodes, the blue node is the parent of the group circle. (b) The groups are positioned around a central node. (c) An illustration of drawing a group circle around a central node.

that a child node belongs to its parent. The lines within a circle also signal a sense of integration. But in order to avoid overemphasizing the lines instead of the nodes, and sometimes to avoid occlusion between lines and nodes, we choose to increase the transparency of the lines[3]. Furthermore, as argued, curved lines can be used to make certain paths in a graph more apparent [181], based on [58], and curved shapes are often reflective of natural objects, giving the observer a pleasant feeling [101], we choose to use Bézier curves instead of straight lines. However, the exact role that curves play in improving the perception of grouping structure and the aesthetics of the visualization is unclear, and beyond the scope of this work.

The groups are then positioned approximately in a circle around the root node that is under focus, as shown in Figure 4.3b. In the initial layout, this top node is the root of the tree. We see that the circumference of each group circle formed by its child nodes is naturally scaled with the number of children, presenting a visual order. Every pair of adjacent group circles are tangent to each other. The CircleTree layout algorithm is detailed in Algorithm 1. The radius r_i of an individual friend node from a group circle c is then approximated by $r_i \approx \pi \cdot r/|c|$, where $|c|$ is the number of friends in c, r is the radius of c. Note that a very large or small m results in an exceptionally small or large r_i. Thus, minimum and maximum radii r_{min} and r_{max} are set to prevent each friend node from being too small to see or too large that it disturbs the visual ordering. When c has few friends, its assigned r becomes small, making $r_i < r_{min}$. After

[3]Note that this intended reduction of opacity does not make the lines difficult to see on a computer screen, but may lead to sub-optimal printing quality.

Algorithm 1 The algorithm for computing the layout of the group circles around a center (x_0, y_0). Note that (x_0, y_0) can be the position of the root or any center of a parent node of a group circle. We also set the maximum angle for each circle to $\pi/2$, which is an empirically derived value to keep the sizes of the generated circles contained within the canvas. For symbols θ, x_0, y_0, x, y, r and R, please refer to the illustration in Figure 4.3c.

Require: the array Arr storing the sizes of the circles.
n = No.Circles, N = No.Friends, $MaxAngle = \pi/2$.
Let the array $Angles$ store the angles θ the circles.
for $i = 0$ to $n - 1$ **do**
    $Angles[i] = 2\pi \cdot (Arr[i]/N)$
    if $Angles[i] > MaxAngle$ **then**
        $Angles[i] = MaxAngle$
    end if
end for
Let the array CS store the tuples (x, y, r).
if n > 1 **then**
    $x = x_0 + |\tan(Angles[0]/2) \cdot R|$
    $y = y_0 - R$, $r = x - x_0$
    $CS[0] = (x, y, r)$
    $totalAngle = Angles[0]$
    for $i = 1$ to $n - 1$ **do**
        $r = |\tan(Angles[i]/2) \cdot R|$
        $s = \sqrt{(r^2 + R^2)}$
        $x = x_0 + \sin(totalAngle + Angles[i]/2) \cdot s$
        $y = y_0 - \cos(totalAngle + Angles[i]/2) \cdot s$
        $CS[i] = (x, y, r)$
        $totalAngle = totalAngle + Angles[i]$
    end for
else
    $CS[0] = (x_0, y_0, R)$
end if
return CS

restoring the overly small r_i to r_{min}, we will likely have relatively large child nodes occupying the entire inner space of c and overlapping with the central parent, which does not make sense to show. Therefore such friend nodes are automatically hidden from sight, instead, the user will only see the grey circular silhouette around the parent to mark the visual area of the group, keeping the visualization clean and ordered, as illustrated in Figure 4.3b.

In the visualization, initially, the user only sees one layer of the tree, as an

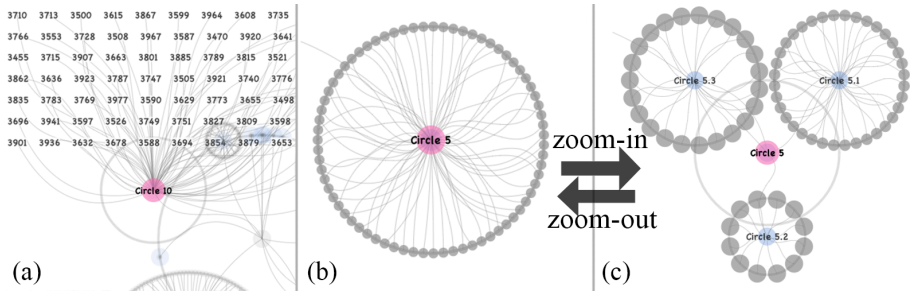

Figure 4.4: (a) Right-clicking a parent node reveals the names of friends in that group. (b) A group of friends before zooming-in. (c) The same group of friends from (b) who are further grouped after zooming-in.

overview, but can further explore it by zooming, panning and enabling text labels. We assume that a user can recall his/her impression of his/her relationship with a friend if he/she sees that friend's name. Therefore, when the mouse hovers over a node, the node is highlighted and the corresponding label is shown, either a friend name or the name of a numbered intermediate node (e.g. "Circle 5" or "Circle 5.3"). Right-clicking on a parent node maps its child nodes (which we call "the focused children") in a grid layout with the names brought into sight. When a grid layout is triggered, we increase the transparency of all the other nodes on the canvas, so as to reduce the interference from irrelevant visual objects, but still keep them visible in the background to maintain a global context, as shown in Figure 4.4a. Clicking (left or right) anywhere other than "the focused children" or another parent node on the canvas will restore the original layout. Right-clicking on another parent node will automatically restore the circular placement of the currently focused children, while shifting focus onto the children of the newly clicked parent node. The user can pan (drag to displace visual objects) to adjust the point of interest. If the starting point of panning is not over a node, the whole tree will be panned. If it is over a node, that node will be panned, along with its child nodes if it is a parent.

We take the current mouse position on the canvas as the "anchor point" for zooming actions. An anchor point $P_{anchor} = (x_a, y_a)$ is the position that is invariant during zooming. A zooming action triggers the following transformation: $rt \cdot \beta \cdot (P' - P_{anchor}) = (P' - P)$, in which $P = (x, y)$ is the position before zooming, $P' = (x', y')$ is the position after zooming, $rt \in \mathbb{R}$ is the value of mouse-wheel rotation provided by the operating system, and $\beta \in \mathbb{R}$ is a constant adjusting the zooming speed. Note that the zooming speeds on X- and Y-axes are the same. It then follows that the scaling factor is $sf = (x' - x_a)/(x - x_a) = (1 - rt \cdot \beta)^{-1}$. During zooming, the radius r_i of each

node is multiplied by sf but further constrained by $r_i \in [r_{min}, r_{max}]$. When the child nodes no longer overlap with the corresponding parents, the hidden child nodes and their names are brought into display with zooming-in. When the user zooms into one circle of friends, we perform a "focus-check" to determine whether to further divide the circle. The "focus-check" assumes a rectangular area, half the width and height of the canvas, with the current mouse position as the center point. Upon the user's zooming-in, the only remaining group circle whose parent node is within this area is found and divided. The newly generated sub-circles are presented if the subgraph corresponding to the circle is divisible according the algorithm [134]. We choose the size of the "focus-check" area such that the user does not need to zoom too deeply or too shallowly to explore sub-circles. Zooming out of the visualization makes the sub-circles from the previously divided circle squeezed and overlapped, which will trigger them to merge back to the singular circle again. This is depicted in Figure 4.4b and 4.4c.

Furthermore, we redesigned the way that the user logged in with his/her Facebook account. More specifically, we changed the tool from a desktop tool (FreeBu#1) to an online application (FreeBu#2). The user no longer needed to copy and paste the access token. Instead, FreeBu#2 provided Facebook's standard login window and the user could log in the tool like other third-party Facebook online applications, as shown in Figure 4.5. This created a concern for the users' data safety. Therefore, we provided the "opt-out" button. In this way, we made it explicit that we would collect the user's data, but the user had the choice of not participating in the data donation.

We also redesigned the way the user could construct his/her own Facebook lists. The user creates a new, empty list by clicking the "plus" button on the top-right corner of the screen. A new rectangle representing the new list will appear aligned on the right of the screen. The user can right-click a list to edit its name. Drag-drop actions put selected friends into a list, as illustrated in Figure 4.6. The user can create an arbitrary number of lists, give any names to the lists and put any friends into the lists. Friend-overlapping between lists is allowed. Mouse-hover over a list brings out the members in that list. Each member is represented as a small rectangle with a removal button (the cross sign), so that the user can edit a list after drag-drop. A list itself can also be removed by clicking the removal button. Once the user is satisfied with his/her custom friend lists, he can then click the submit button to create the corresponding Facebook friend lists in his/her account. These actions give much more flexibility in list composition than FreeBu#1.

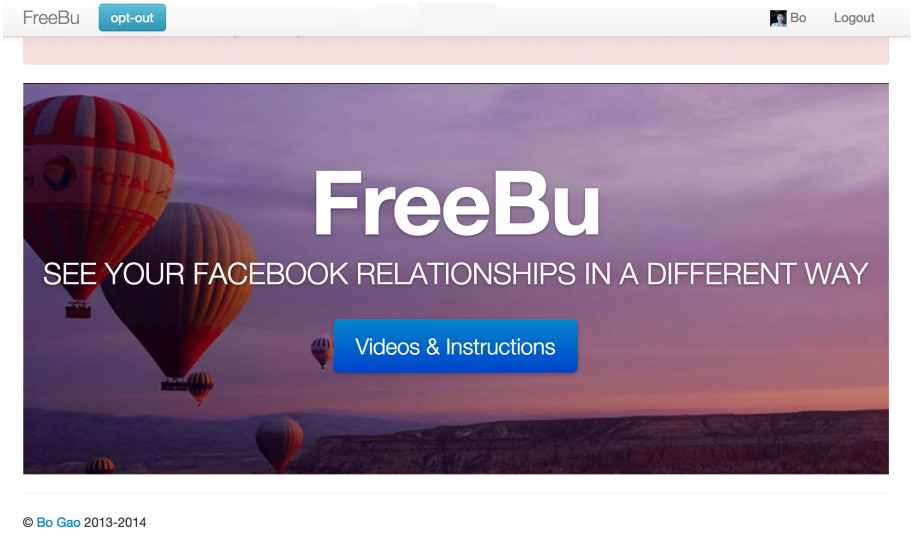

Figure 4.5: The homepage of FreeBu#2

4.4 Common Regretted Posts (RQ2)

Our user study consists of two parts. The first part is the solicitation of the participants' common scenarios of regretted posts. The second part is where we let the participants make "visibility decisions" based on their regretted posts with either the HMOD or the FSL grouping strategy. In this section, we define the term "visibility decision" and describe the part of the user study that investigates the common scenarios of regretted posts from the participants.

4.4.1 Visibility Decision

We use the term Visibility Decision to refer to a user's binary decision on whether a post is visible to an individual friend in his/her egocentric networks. A post can be anything that a user uploads or shares in an OSN, e.g. a status update, a (re)tweet, a photo, a comment or an article shared, etc. Friend grouping can facilitate users' visibility decisions. The user decides the visibility of a post directly based on friend groups rather than individuals. In other words, when the user sees a group, assuming his/her previous familiarity with the group, he can skip the serial browsing that examines individual friends in this group, and determine the visibilities of the post towards those friends on a group level. There are two exceptions in which the user does not directly deploy

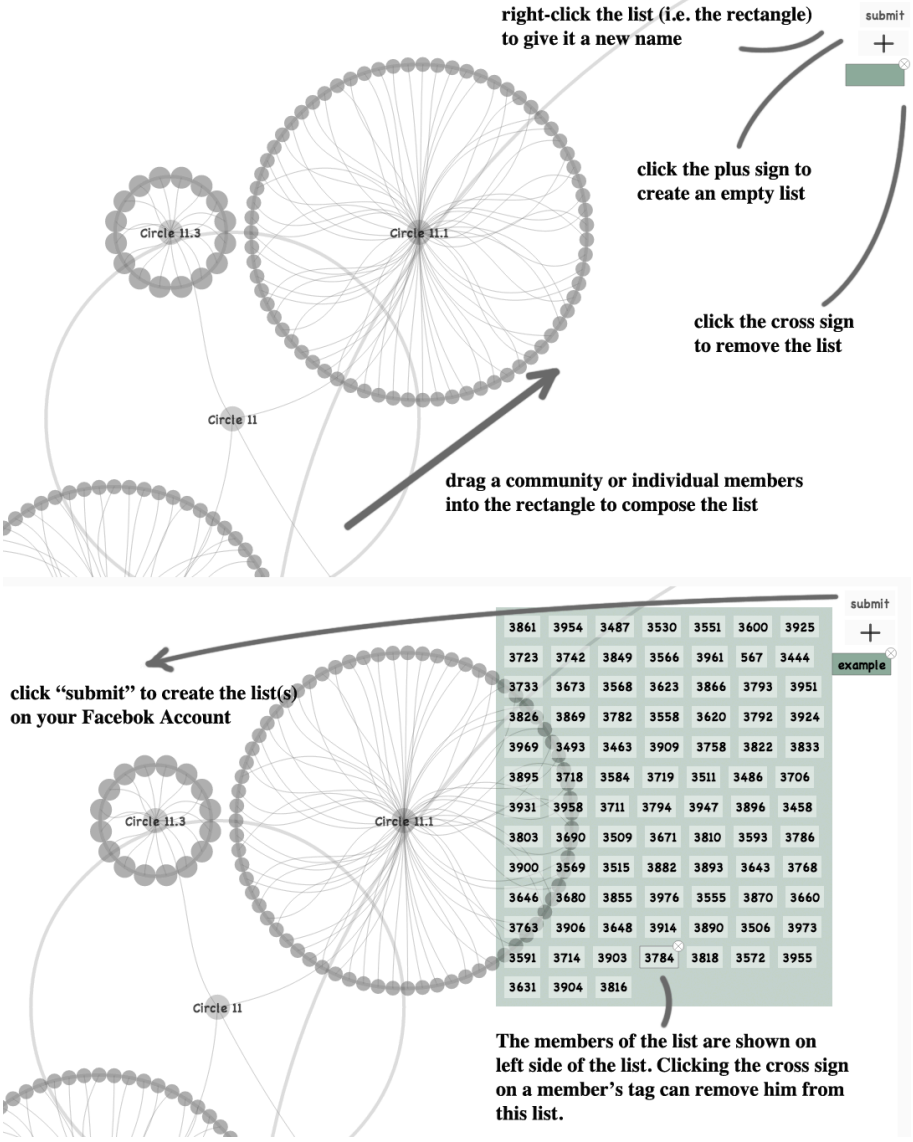

Figure 4.6: The drag-drop actions to compose a custom Facebook friend list

the groups in his/her visibility decisions. *First*, certain posts are too privacy sensitive, i.e. it has become a complete regret, or not sensitive at all. In both cases, a binary decision becomes unary, and all user's friends are considered as

one group. *Second*, when the number of friends who are or are not supposed to see a post (e.g. one, two or three) is significantly smaller than the number of friend groups shown to the user, then checking the groups requires more effort than just doing a standard search, e.g. typing friend names in a search box. Thus, it is no longer necessary to use groups. However, we shouldn't completely disregard friend grouping in such situation, because it can raise the user's awareness about his/her friends, which can help the user spot "unexpected" or "surprising" friends, which then becomes useful for the user to make visibility decisions.

It is important to note that in order to make visibility decisions for a post at the very beginning, the user needs to go through all the friends, regardless of the form of presentation, either simple textual lists on a paper or complex visualizations. The benefit of a (good) friend grouping follows after the user's initial contact and familiarization with the generated groups. In other words, the user has made the connection between members and their corresponding group. A group is represented by a token, which can be a shape, a descriptive phrase, or the name of a member from this group, etc. This linkage information is stored in the user's long-term memory. The members can be recalled when the user just sees the group token. In such a way, the user bypasses the serial browsing of each individual member, and directly utilizes a group. However, in order to determine which visual cues in the visualization act as the group tokens, and to confirm that the user indeed utilizes the token to bypass individual friends in a group, further studies need to be carried out.

4.4.2 Soliciting Regret Scenarios

We recruited 16 participants (three females), 25-45 years old, from eight countries. Among them were Ph.D students, company employees and graduate students. Each participant was asked to identify his/her regretted posts. A regretted post is a post in the past which the user felt was shown to unintended friends.

Though recent studies have investigated regrets in OSN from different aspects [127, 178], we chose to let the participants explicate their own regrets, as it is easier for a person to make visibility decisions based on his/her own experience. We collected the posts in face-to-face interviews with the participants. We emphasized the difference between complete and partial regrets. A complete regret meant that the post was supposed to be seen by no one. A partial regret meant that the participant did not mind his/her post being seen or intended his/her posts to be seen by some of the friends, but failed to block the other undesired friends. Since a complete regret entails concealing the corresponding post completely, which would render a visibility decision trivial,

Table 4.1: Participants' Regretted Posts

	Categories of Regretted Posts	Frequency
(1)	sensitive photos causing embarrassment or awkwardness	8
(2)	other photos for a specific group of friends	9
(3)	sensitive topics involving emotional expressions	7
(4)	sensitive topics involving nasty jokes	12
(5)	other topics for various specific situations	12

we guided the participants to only think of partial regrets. Each participant was encouraged to think of at least three posts. A post needs to be specific enough to let the participant define its visibility towards each friend. In total, 48 posts were collected; each participant contributed three personal posts on average. We found that photo-related posts were mentioned frequently, thus making a distinction between photos and topics. Topic-related posts include status updates, web-link sharing and comments.

We recorded the participants' regretted posts and manually classified them into five categories, as summarized in Table 4.1.

- The *first* category covers the posted photos that cause embarrassment or awkwardness, typical examples are "drunk party" photos. There are also the photos showing the participant together with some particular person(s), e.g. ex-boy/girl-friend, for which the participant feels the need to hide the photos from some friends.
- The *second* category covers the photos that are less sensitive in terms of embarrassment or awkwardness, but still in need of visibility control. For example, some photos may be so intimate that the participant only wants to show them to his/her family and best friends. Some photos were taken at a event with a specific group of people, only to whom, as participants argue, the photos should be made visible. More than a third of the posts are photo-related.
- The *third* category covers the topic-related posts that involve explicit self-expression, including strong opinions and emotional expressions, such as venting negative emotions. Of the seven posts in this category, six are about venting or expressing negative opinions, which the participants felt should be avoided in the future, for those posts may harm one's image if disclosed carelessly.

- The *fourth* category covers the sensitive topics that are less self-involved, but more about the intrinsic sensitive nature of the content of the posts, including politics, religion, sex, race and/or nasty jokes. It is interesting to see that nine out of the twelve posts in this category are about inappropriate jokes. For example, several participants reported that they posted something they believed sarcastically humorous, but in hindsight, they thought it was not wise to expose those posts publicly, as some friends may not understand the humour, or even be offended by it.
- The *fifth* category covers the relatively less sensitive topic-related posts, which nonetheless need visibility control. For instance, it may not make sense to show the posts to the friends who do not speak the language in which the posts are written.

4.5 Comparing Two Grouping Methods (RQ3)

To examine the difference between two friend grouping strategies, there needs to be one common User Interface (UI). Given that a Facebook user usually has hundreds of friends, naively using "pen and paper" to elicit the visibility decisions from the participants may weary them. Bearing this in mind, we decide to let the participants operate with the same computer-based UI. For users making visibility decisions with such an interface, we need two basic functions: *First*, browsing is applicable at both group and individual levels. *Second*, making a decision is applicable at both group and individual levels.

In this section, we describe the part of the user study that compared the usefulness of two grouping strategies to Facebook users, namely FSL and HMOD. The same participants as described in Section 4.4 put into A/B testing, i.e. they were equally divided into two groups and each group has eight participants. We named two groups directly after the corresponding strategy abbreviations – FSL and HMOD.

Both groups used the visualization interface detailed in Section 4.3, but with different grouping strategies, as their names suggested, namely the Facebook smart lists and the hierarchical, modularity-based community detection strategy used in an interactive way. Because the former was not a complete grouping, the friends of a participant that were not in any smart list were put together as one other group.

Our assumption in the user study is that users utilize categories of friends (denoted as C_u) to make a binary visibility decision. We denote the communities that HMOD and FSL produce as C_{HMOD} and C_{FSL} respectively. C_{HMOD} is

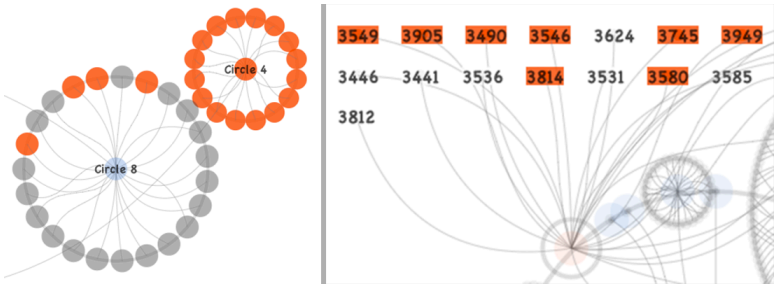

Figure 4.7: Participants can determine a post's visibility to each friend individually by clicking friend nodes or collectively by clicking parent nodes in the centers of group circles.

the result of the interactions between a user and HMOD, with the CircleTree visualization interface. C_{FSL} is the set of non-hierarchical circles of friends constructed from the user's Facebook smart lists, with one extra circle containing the friends who are not in any of the smart lists. Our hypothesis is that, for users' visibility decision-making, C_{HMOD} coincide with C_u, more than C_{FSL}.

We asked each of the participants to make visibility decisions for each of their regretted posts. Each group has 8 participants and 24 posts. As illustrated in Figure 4.7, when a participant thinks a friend can see the post, he clicks on the corresponding friend node, the color of which changes to indicate that the post is now visible to the clicked friend. Clicking on the parent node of a group circle toggles every child node's color, or further descendants if some child nodes are already divided by a zooming-in action. The participants could work at their own pace until they were satisfied with their decisions.

We use binary entropy to evaluate the effectiveness of the two approaches for users making visibility decisions. $Entropy(post) \in [0, 1]$ (Equation 4.1) calculates the information content (in bits) needed to determine whether a member in a circle can see a post. C is a set of circles of friends, and $c \in C$ generated by HMOD or FSL. $V_{c,post}$ is the number of the friends to whom $post$ is visible in the circle c. N is the total number of friends (including duplicates if circles overlap) in all the circles. $Entropy(post) = 1$ means that on average, in one circle, half the circle can see the post while the other half cannot. This indicates that the given set of circles is unhelpful for the user to make visibility decisions on a group-level, by taking the circles holistically into account. $Entropy(post) = 0$ means that for each circle, the friends in the same circle have the same visibility access to the given post. That is, every circle can be fully utilized by the user to make visibility decisions. The CircleTree visualization in the group HMOD is analogous to a binary classification tree.

Users try to use this tree to make visibility decisions. A"pure" circle in terms of visibility decisions is helpful, since such a circle can be considered as a whole. The initial circles are divided until they are indivisible according to the graph modularity or they are pure. Then the sub-circles are used to calculate entropy scores.

$$Entropy(post) = \sum_{c \in C} \frac{|c|}{N} Entropy(c, post) \text{ with}$$

$$Entropy(c, post) = -\frac{V_{c,post}}{|c|} \cdot log_2 \frac{V_{c,post}}{|c|} - \frac{|c| - V_{c,post}}{|c|} \cdot log_2 \frac{|c| - V_{c,post}}{|c|} \tag{4.1}$$

Another aspect of a set of visibility decisions for a user's post is its imbalance. That is, the number of friends who can see the post is significantly different than those who cannot see the post. Let V_{post} be the total number of friends who can see the post and $\alpha = min(V_{post}, N - V_{post})$. When α is rather small, e.g. one or two, $Entropy(p)$ can be low almost regardless of which grouping method is used. In such cases, while a grouping may still be useful for the participants to browse friends, but it is likely to be less effective for making visibility decisions than the participants just typing individual friend names to search for them in real-time. We know that the average number of friends of each participant is 194. All the 48 posts (24 posts for each group) have $\alpha > 1$ and $\overline{\alpha} \approx 34$. Within these posts, there are 38 posts (19 posts for each group) with $\alpha > 5$ and $\overline{\alpha} \approx 42$. Table 4.2 shows the average Entropy scores in group HMOD and FSL for $\alpha > 1$ and $\alpha > 5$. Group HMOD achieves lower entropy than group FSL in both cases. This suggests that the circles generated by the hierarchical modularity-based method are taken more holistically into consideration than Facebook smart lists by the participants to make visibility decisions. In other words, it happens more often that a circle in the HMOD group, rather than one in the FSL group, is marked unanimously as the people who "can see" or "cannot see" a post. We can also see that raising the α level indeed increases the average entropy scores in both groups, but the increase is more apparent in group FSL ($\approx 22\%$) than in group HMOD ($\approx 10\%$).

We test the statistical significances of the differences between the entropies from HMOD and FSL. It is, however, less straightforward to compare the two, because the entropy scores yielded in group FSL are from a different set of participants, with a different set of egocentric networks and posts. Nevertheless, it is possible to perform an approximate comparison by a pessimistic pair-wise matching. We first calculate the pair-wise squared entropy differences between FSL and HMOD/MOD, deriving a cost matrix, with which, we match the

Table 4.2: Entropy scores for group FSL and group HMOD, with $\alpha > 1$ and $\alpha > 5$.

	FSL	HMOD
$\alpha > 1$ (24 posts)	**0.46**	**0.20**
$\alpha > 5$ (19 posts)	0.56	0.22

entropies in the two groups via the linear assignment [129] to minimize the sum of the pair-wise differences (so as to minimize the difference between the two models). Based on the resulting pair-wise matches, we perform the t-tests. It then follows that, in comparing HMOD and FSL, the t-statistic is 9.146 for $\alpha > 1$ and 12.810 for $\alpha > 5$. The t-statistics reject the corresponding null hypotheses with two-tail Confidence Interval $(CI) = 99.9\%$. It is evident that HMOD is significantly better than FSL.

4.6 Conclusion

Informed by the limitations of FreeBu#1, the human cognition argument for friend-grouping and an overview of the ways of visualizing hierarchical data, we designed FreeBu#2. We then used its visualization interface to conduct the user study that investigated the common scenarios of regretted posts for Facebook users and compared the two grouping strategies FSL and HMOD. The result showed that HMOD helps Facebook users more conveniently make visibility decisions than FSL.

Furthermore, from this user study, we gain more insight into users' privacy decision-making process in online posting.

First, it is evident that categorical thinking is used when users make binary visibility decisions.

Second, graph-modularity-based friend communities assist users more efficiently for such decisions than the profile-attribute-based Facebook smart lists. This implies that the former produces the communities that fit the categories of friends that a user has in mind, more than the latter. We examine another state-of-the-art community detection algorithm for egocentric networks in comparison with the modularity-based algorithm in Chapter 5 and discuss the implications of the results.

Third, the essence of categorical thinking is to reduce the cognitive load. If there are too many information objects (in our case, online friends),

hierarchical categorization supports the users' visibility-decision-making. When the resolution limit of MOD is reached, the sub-circles can be of more help to the users. The results also give us guidance in designing information visualization systems – categorization and abstraction are important for users to process large amounts of information.

Chapter 5

Investigating Community Detection in Ego Networks

From previous chapters, we know that it is important to categorize one's online friends in order to make sense of the online social context and make decisions. Though the modularity-based community detection technique used in the grouping tool has shown to be quite useful, we do not know how accurately the machine-generated groups match those that are manually created by OSN users on a large scale, and how this particular technique performs compared with other methods. There exists abundant and different techniques for community discovery on graph (nodes and edges) and/or feature data (attribute values on nodes and/or edges). Leskovec and McAuley [124] proposed a Generative Model for Friendships (GMF) that leverages both graph and feature data to detect communities on egocentric OSNs. They also conducted comprehensive experiments to show that GMF outperformed the other major community discovery techniques on three ground-truth datasets from Google+, Twitter and Facebook. However, Newman's modularity-base community detection method (MOD) was missing from this comparison. Furthermore, we also want to investigate whether there is room for improvement on the MOD technique for OSN community detection. Therefore, we corresponding research questions are:

RQ1: Which method produces the circles that match OSN users' manually created circles better, GMF or MOD?

RQ2: How can we improve MOD so that the circles it produces match OSN users' manually created circles better?

We first introduce the two community detection methods in Section 5.1, and the datasets in Section 5.2. We then address the two questions in Section 5.3 and Section 5.4 respectively.

5.1 Two Community Discovery Models

From our preliminary user study, we know that, in order to make sense of the friends in one's online social life, it is important to categorize them, either for the ease of processing and memorizing friends' information, or as an efficient means for making decisions. Given the large number of friends that an OSN user usually has, automated community detection can be very helpful not only as the basis for visibility decisions, as investigated in the previous section, but also for other tasks in online contact management, such as simply keeping an overview, sorting incoming messages, etc. In this section, we examine community detection algorithms and their relationship with real-life social groups. We compare two models for community detection in egocentric networks: the graph-modularity-based model (MOD) using eigenvalue decomposition [133] and the Generative Model for Friendships (GMF) [124].

Modularity is the number of edges falling within groups minus the expected number in an equivalent network with edges placed at random [134]. Larger modularity value suggests more obvious community structure in the graph. There exists abundant and different techniques that optimize the modularity of a graph. We chose to implement Newman's spectral optimization algorithm that iteratively bisects a given graph using the eigenvectors of the modularity matrix. This approach is generally more accurate than the techniques such as greedy methods and external optimization, and less computationally expensive than global optimization approaches such as simulated annealing [60]. We also implement vertex-moving to improve the final modularity score, as proposed in [134]. Intuitively, in each bisection of the input (sub-)graph, "vertex-moving" moves one vertex at a time, from one (sub-)community to the other, if the modularity is increased, it makes this move permanent. The average, combined complexity of this algorithm is $O(N^2 log\ N)$.

GMF is a recently proposed community detection model that leverages both the friend-profile features and the friend-graph structure in an egocentric network [124]. The resulting communities have the following properties: (1) the friends in the same communities have common features, such as education, work; (2) different communities may emphasize different features; (3) the communities may overlap. GMF has been evaluated against the ground-truth communities from three egocentric-network datasets (as described in Section

Table 5.1: Three ego-network datasets summarized, from left to right: $\overline{|V_{circles}|}$ is the average number of friends from a user's ground-truth circles, $\overline{|V_{edges}|}$ is the average number of friends from a user's friend graph, $\overline{|C|}$ is the average number of a user's ground-truth circles, $\overline{|c|}$ is the average ground-truth circle-size, $\overline{No.Comms.P}$ is the average number of ground-truth circles to which a friend belongs.

ego-networks	$\overline{\lvert V_{circles}\rvert}$	$\overline{\lvert V_{edges}\rvert}$	$\overline{\lvert C\rvert}$	$\overline{\lvert c\rvert}$	$\overline{No.Comms.P}$
Facebook (10)	298	423	19.3	26	1.6
Twitter (909)	36	134	4.4	12	1.4
Google+ (129)	304	1948	3.6	135	1.6

5.2), and compared with eight baseline models – Mixed Membership Stochastic Block Models, Block-LDA, K-means clustering, Hierarchical Clustering, Link Clustering, Clique Percolation, Low-Rank Embedding and Multi-Assignment Clustering (as elaborated in [124]). It was demonstrated that GMF generated more accurate communities than the baselines.

5.2 Three Egocentric-network Datasets

The three datasets were collected from Facebook, Twitter and Google+, which are available online[1]. We downloaded these datasets, removed empty files, and discarded the ego-networks whose ground-truth circle(s) contains just one friend. Finally we obtained 10, 909 and 129 ego-networks from Facebook, Twitter and Google+ respectively, which we use for our experiments. Note the data is a subset of the data used in [124]. Each ego-network includes the user's and the friends' profiles, the friend graph and the set of manually constructed circles by the user. For the Twitter and Google+ friend graphs, we ignore their directivity as MOD runs on undirected graphs. We denote the complete set of friends as V, the friend nodes retrieved from the user's ground-truth circles in an ego-network as $V_{circles}$, the friend nodes retrieved from the user's friend graph as V_{edges}, a ground truth circle as c, the set of ground-truth circles as C, an algorithm-generated circle as c' and a set of algorithm-generated circles as C'. The three datasets are summarized in Table 5.1. We see that $\overline{|V_{circles}|} < \overline{|V_{edges}|}$ for the three datasets, since $V_{edges} \subseteq V$, it indicates that $V_{circles} \subset V$. Moreover, we observe that overlapping ground-truth circles are common, but also limited such that a friend is usually assigned to less than two circles.

[1] `https://snap.stanford.edu/data/index.html#socnets`

5.3 Performance of GMF and MOD (RQ1)

We follow the same method and metrics in [124] to evaluate how well a set of generated circles C' match the user's manual circles C. Balanced Error Rate (BER) [36] and F1 scores are used to measure the matches of circles, as defined in Equation 5.1 and 5.2. We use $RBER(c, c')$ to refer to $1 - BER(c, c')$. In order to determine which $c' \in C'$ corresponds to which $c \in C$, we perform a linear assignment using the Hungarian Algorithm [129] to maximize the sum of the pair-wise $RBER$ or $F1$.

$$BER(c, c') = \frac{1}{2}\left(\frac{|c \backslash c'|}{|c|} + \frac{|c' \backslash c|}{|V_{circles}| - |c|}\right) \tag{5.1}$$

$$F1(c, c') = 2\frac{|c \cap c'|}{|c| + |c'|} \tag{5.2}$$

We ran GMF[2] and MOD on the ego-networks that only included the friend nodes from ground-truth circles, so that we could compare C and C'. The reason that we ran GMF again instead of directly using its original result was because of the incomplete ego-network data that we could download and some trivial data (e.g. an ego-network containing only one friend) that we discarded afterwards. As such, both GMF and MOD were run on the subsets of the ego-networks that were described in [124], namely 10 Facebook, 909 Twitter and 129 Google+ ego-networks instead of 10, 1000 and 133 ego-networks. Due to the complexity of the algorithm (with the worst case complexity $O(N^3)$, N being the number of friend nodes in an ego-network), we ran GMF for each ego-network with selective K values (the number of communities), $K = 3, 5, 7$ and 9 respectively. Then we select the K value that corresponds to the highest average $\overline{RBER}$ or $\overline{F1}$, and match C and C' for each ego-network via linear assignment. As for MOD, K is automatically derived in the process of modularity maximization. The results are summarized in Table 5.2 and 5.3. Note that while certain K of GMF achieves the highest $\overline{RBER}$, it does not necessarily mean this K corresponds to the highest $\overline{F1}$. Thus we have two different sets of combinations of Ks with respect to the $\overline{RBER}$ and $\overline{F1}$ measures. The columns $|\overline{C'}|$ and $\overline{No.Comms.P}$ in Table 5.3 are based on the average values of these two sets of Ks.

We denote the GMF algorithm that was run on the original ego-network datasets, with a full range of K values checked, as GMF0. This is to differentiate it

[2]The code can be downloaded from the author's web page: `http://i.stanford.edu/~julian/`. We used the default parameters in the code with different K values.

Table 5.2: The comparison between the results of GMF running on the subsets with four K choices (white columns) and the original sets (gray columns) of the ego-networks: Facebook (Fb), Twitter (Tw) and Google+ (Gp).

GMF	Fb(10)	Fb(10)	Tw(909)	Tw(1000)	Gp(129)	Gp(133)
$\overline{RBER}$	0.83	**0.84**	**0.77**	0.70	0.65	**0.72**
$\overline{F1}$	0.53	**0.59**	0.32	**0.34**	0.24	**0.38**

Table 5.3: The results of running GMF and MOD on the three subsets of ego-networks. The gray sub-columns are the results for GMF, the white ones are for MOD. $\overline{|C'|}$ is the average number of generated circles, $\overline{|c'|}$ is the average size of each generated circle, $\overline{No.Comms.P}$ is the average number of circles to which each friend belongs.

ego-networks	$\overline{RBER}$		$\overline{F1}$		$\overline{\lvert C'\rvert}$		$\overline{\lvert c'\rvert}$		$\overline{No.Comms.P}$	
Facebook	0.83	**0.86**	0.53	**0.67**	3.3	7.0	90	41	1.5	1
Twitter	0.77	**0.81**	0.32	**0.68**	5.2	3.0	7	12	2.7	1
Google+	0.65	**0.75**	0.24	**0.62**	6.9	3.1	44	98	3.8	1

from the GMF model that we ran on the subsets, with the four K values checked. From Table 5.2, we notice that the $RBER$ and $F1$ scores of GMF on the Facebook and Google+ datasets are smaller than those of GMF0, and the $RBER$ and $F1$ scores of GMF on the Twitter dataset are comparable to or higher than those of GMF0. The relatively large performance difference on Google+ is due to the limited choices of K in GMF. From Table 5.3, we see that MOD fully outperforms GMF on $RBER$ and $F1$ measures.

5.4 Multi-membership Modularity-based Community Discovery (RQ2)

From Table 5.3, we can also see that MOD generates the $\overline{|C'|}$ that is closer to the ground-truth as shown in Table 5.1. We also know that though overlapping circles are common in the ground-truth, one friend is rarely put into more than two circles, whereas GMF on Twitter and Google+ generates the circles that have $\overline{No.Comms.P}$ equal to or larger than three, which led to its relatively low performance on these datasets. However, a significant limitation of MOD is that it produces non-overlapping communities, while it is obvious that OSN users construct overlapping circles by themselves. Thereby, we propose an

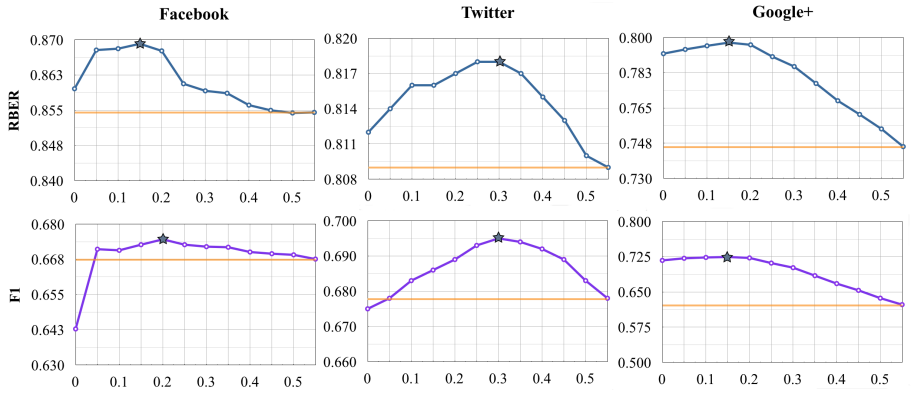

Figure 5.1: The $RBER$ and $F1$ performances of MMOD with different θ_{EB} values. The baselines are drawn to indicate the corresponding MOD performances and the stars are to mark the optimal θ_{EB} points.

extension of MOD that allows multiple circle memberships, which we call Multi-membership Modularity-based community detection, shortly as MMOD. We define a metric we call the External Belongingness (EB as in Equation 5.3), in which $neighbors(v, c')$ is the number of neighbors (one hop away on the friend graph) of a given friend v in an external circle c', $degree(v)$ is the degree of v. c' is external to v if $v \notin c'$. We first run MOD to derive a set of non-overlapping circles. Then for each friend, we obtain a list of external circles (the circles to which the friend does not belong) with the corresponding EB scores. We subsequently check the highest EB score for each friend, if it exceeds the previously defined θ_{EB}, the friend is assigned to the corresponding external circle. In this way, we obtain a set of overlapping circles with some friends belonging to two circles. However, it remains the question of how to select θ_{EB}. We run MMOD with different $\theta_{EB} \in [0, 0.5]$ with the step size 0.05. Then we match the respective overlapped C' with C, the performances are plotted in Figure 5.1. In each plot of Figure 5.1, the last point is the average $RBER$ or F1 score from MOD, through which a straight horizontal line is drawn to indicate baseline performance. The point with the highest performance is marked with a star.

$$EB(v, c') = \frac{neighbors(v, c')}{degree(v)}, v \in V, v \notin c' \tag{5.3}$$

From Figure 5.1, we can see that the performances of MMOD are generally better than those of MOD. The curves also follow the similar trend that increases till some particular θ_{EB} and drops. Around $\theta_{EB} = 0.5$, rarely any friend nodes can

Table 5.4: The results of running MMOD on the three subsets of ego-networks. $\overline{|C'|}$ is the average number of generated circles, $|c'|$ is the average size of each generated circle, $\overline{No.Comms.P}$ is the average number of circles to which each friend belongs.

ego-networks	$\overline{RBER}$	$\overline{F1}$	$\overline{\lvert C'\rvert}$	$\overline{\lvert c'\rvert}$	$\overline{No.Comms.P}$
Facebook	0.87	0.67	7.0	52	1.3
Twitter	0.82	0.70	3.0	18	1.5
Google+	0.80	0.73	3.1	166	1.7

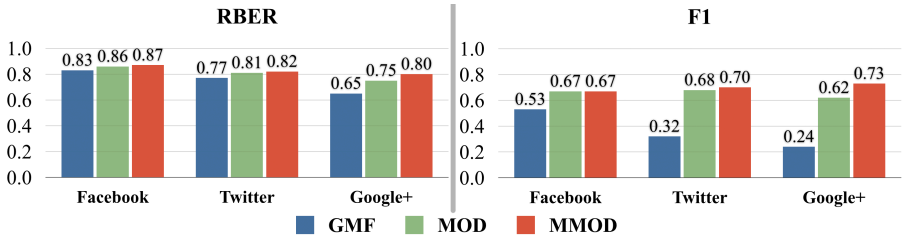

Figure 5.2: The overview of the performances of GMF, MOD and MMOD.

be found in external circles, thus the performances regress to be close to MOD's. We also find that the optimal threshold θ_{opt} values for Facebook and Google+ data are similar, which stay around 0.15 for both $RBER$ and $F1$, whereas for Twitter data, this value is 0.35. The $RBER$ and $F1$ scores of MMOD at these θ_{opt}, along with other results (the same columns as Table 5.3) are summarized in Table 5.4, from which we see that MMOD fully outperforms MOD, and that the $\overline{No.Comms.P}$ values are very close to those of the ground-truth datasets. The better results on MMOD also have the implication that people indeed tend to put the friends who are the connectors or hubs in the ego-network into different circles at the same time. We summarize the performances of GMF, MOD and MMOD in Figure 5.2.

We also observe that θ_{opt} empirically correlates with the average size $\overline{|V_{circles}|}$ of an ego-network, which is around 300 on Facebook and Google+, and 30 on Twitter. For instance, we can describe this relation with Equation 5.4. If we consider the MMOD-generated circles match the user's manual circles better (indeed, the $RBER$ rates are close to or well above 0.8), the relation in Equation 5.4 suggests that, on the one hand, users tend to manually create less overlapped circles when they have fewer friends. On the other hand, θ_{opt} decreases exponentially slower than the number of one's friends increases, which means that on a relatively large scale (e.g. $|V_{circles}| \in [100, 1000]$), given that

Table 5.5: The **p** values of the ANOVA for GMF, MOD and MMOD, of both $\overline{RBER}$ and $\overline{F1}$measures, on the datasets of Facebook, Twitter and Google+ respectively.

	Facebook	Twitter	Google+
$\overline{RBER}$	.43	$< .001$	$< .001$
$\overline{F1}$	.13	$< .001$	$< .001$

ego-networks are often sparse [172, 128], users' θ_{opt} for allowing a friend to be in multiple circles remains similar ($\theta_{opt} \in (0.12, 0.20)$ approximately). However, in order to accurately capture the relationship between the number of friends and the optimal threshold, we need a further investigation. It may involve other potentially correlated parameters, more sophisticated models and more data, which is beyond the scope of this work. Equation 5.4 is manually derived based on the observations from Table 5.1 and Figure 5.1. It serves as an intuitive guidance for determining θ_{opt}.

$$|V_{circles}| = 3 \times 10^{(\frac{0.3}{\theta_{opt}})} \Longleftrightarrow \theta_{opt} = \frac{0.3}{lg|V_{circles}| - lg3},\ \theta_{opt} > 0 \tag{5.4}$$

We perform ANOVA (ANalysis Of VAriance) to compare GMF, MOD and MMOD on the three datasets. The **p** values are summarized in Table 5.5. We can see that the **p** values on the Facebook dataset are rather high, and the **p** values on the other two datasets are low ($\mathbf{p} < .001$). This means that the variance between the three models is not significant on the Facebook dataset, but very significant on the Twitter and Google+ datasets (in fact, the F-statistics on these two datasets approach the ends of the corresponding F-distribution curves.) In Figure 5.2, the observed differences were statistically significant for both $\overline{RBER}$ and $\overline{F1}$ on the Twitter and Google+ datasets (all $\mathbf{p} < .001$ for one-way ANOVAs), but not for the Facebook dataset ($\mathbf{p} = .43$ for $\overline{RBER}$ and $\mathbf{p} = .13$ for $\overline{F1}$). The latter may be a result of the small sample.

Note that the result of the performance comparison among GMF, MOD and MMOD only applies to the three ego-network datasets at hand. Due to the scarcity of such ground-truth data, the effect of MMOD on a more general level is yet to studied.

5.5 Discrepancy between Predicted and Manual Circles

Though a community discovery algorithm can predict reasonably good circles, it is unlikely that it can make a perfect prediction. This attributes to the fact that manual circle-creation process is inherently subjective, and varies on the same person for different purposes. The ground-truth circles of the ten Facebook users that we used in our experiments were obtained by a Facebook app[3], in which the user entered comma-separated category labels for each friend. Existing labels could be reused by a selection from a drop-down box. Each label represented a circle to which a friend belonged. The text cue for entering the label(s) for each friend **Fr** was "I know **Fr** because ..." followed by the label-entering text-field. In another exercise [47] of friend-grouping, the groups (i.e. circles) were constructed by "card sorting". The name of each friend of a participant's was printed on a paper card. Several cards were randomly selected and spread on a table, the participant was then asked to assign the rest of the cards to the selected ones to form groups. We can see that the Facebook app friend-grouping exercise encourages more overlapping circles to be created than the card-sorting exercise.

Different user interfaces may directly reflect intrinsic and systematic differences on a functional level, rather than on a perceptual level. Facebook provides a social platform mainly for mutual friends – two people become friends when one "accepts" the other's "friend request". The friends of friends are recommended if the user wants to add more friends. Twitter and Google+ implement a "follower-followee" mechanism, which means a friendship is not necessarily reciprocal. On Twitter, the user clicks the "follow" button to follow a "friend", every newly followed friend is not necessarily put into a friend list (i.e. a circle), whereas on Google+, the "follow" button becomes the "add" button, and every newly followed friend has to be added into one of the existing circles or a new one. This is an important reason that the number of friends in Google+ circles is much more than that in Twitter lists, as shown in Table 5.1. We see that people create circles differently under different circumstances, consciously or unconsciously. It is therefore important to create interfaces that help users gain insights about their Ego-network friendships from different aspects, and let them form their own friend circles with more informed decisions.

[3] `https://www.facebook.com/apps/application.php?id=201704403232744`

5.6 Conclusion

In this chapter, we compared two state-of-the-art community discovery models: the modularity-based model (MOD) and the Generative Model for Friendships (GMF). The corresponding two algorithms were run three datasets of ego-networks from Facebook, Google+ and Twitter. The generated communities were compared with the ground-truth circles created by the users. We find that MOD matched the ground-truth more than GMF in terms of Reversed Balanced Error Rate (RBER). We further extended MOD to allow overlapping communities to be generated by introducing a threshold called External Belongingness. The extension outperformed MOD. Through both the comparison and the extension, we found that the MOD approach, which was purely graph-based, can approximate users' manually created circles very well. And leveraging the link information in an ego-network graph to produce overlapping communities also proved to be effective.

Chapter 6

Extending and Evaluating FreeBu

To recap, as studied in Chapter 3, 4 and 5, we understand that grouping friends is important for OSN users to manage online contacts and make privacy-related decisions. A carefully designed community detection algorithm can produce reasonable friend circles that match users' manual circles, but this matching is hardly perfect due to the subjectivity and diversity of friend grouping. To close the gap between computer-based grouping and human grouping, tools need to be developed. These tools leverage interactive/exploratory visualizations and community detection algorithms. The goals of these tools are:

1. visualize users' online relationships in structured ways,
2. enable users' interaction with the structures to create friend groups,
3. encourage users to reflect upon their online relationships.

We have developed FreeBu#1 (Chapter 3) and FreeBu#2 (Chapter 4) to help Facebook users in these regards. However, these tools addressed only part of the information on users' online friends, with limited interactions. We extended FreeBu#2 with three more views that supplement the current circle view, so that the user can inspect his/her ego-network from different perspectives. We call the new version FreeBu#3.[1] In this chapter, we detail the development of FreeBu#3, investigate the affordances of the tool for its users, how the users interact with each visualization and what we could learn from these interactions.

[1] For an overview of the different versions, refer to Appendix C.

6.1 Related Work

As the user study in Section 3.4 showed, people considered attributes such as education, work, interests, language, age, location, etc. when they were grouping the people they know. The user study in [47] showed similar results. This study recruited 18 young adults (age 17-23, 10 males) from Gent, Belgium as participants. For each participant, 100 of his/her Facebook friends' names were randomly selected, and the participant was asked to group these friends in a series of sessions and give the reasoning behind the groupings. Five grouping strategies were identified, as listed below. Note that these strategies are not mutually exclusive.

Shared-community strategy includes education, family, interest/hobby, workplace, location, etc. which cover most attribute-based grouping criteria identified in our previous user study (Section 3.4). In order to avoid confusion with the word "community" in community detection/discovery, we refer to this grouping strategy as Shared-attribute strategy.

Inner-circle strategy refers to the way that a participant categorizes the friends based on whether they know each other. This strategy coincides with the modularity-based community detection method that we adopted in FreeBu.

Mutual-friend strategy refers to the way that a participant uses "one Facebook friend as a common denominator to categorize others. This common denominator, usually, was a closer friend than the people who were placed within this category. The latter were perceived as acquaintances."

Contact-type strategy refers to the grouping strategy using the type of contact that a participant has with his/her friends, such as best friends, drink buddies, etc.

Evaluative strategy refers to the way of grouping that is based on how a participant evaluates others, such as the people I like, the people I respect, the snobs, etc. Both Contact-type strategy and Evaluative strategy are closely related to the "connection-based grouping" that was identified in our previous user study (Section 3.4).

Furthermore, the literature for visual analytics and information visualization [92, 189, 156] unanimously emphasized that presenting multiple aspects and providing multiple perspectives are essential for visualizing large and complex data. Informed by the grouping strategies and data visualization literature, we

decided to enrich FreeBu#2 with additional visualizations/views that emphasize the relevant information for the grouping strategies. And we call the extended tool FreeBu#3.

6.2 Research Questions

Based on our previous work and related literature, we have the following research questions:

RQ1: How do we design the visualizations of FreeBu#3 that correspond to users' different friend-grouping strategies?

RQ2: How do users perceive the values of FreeBu#3?

RQ3: How do users interact with FreeBu#3?

To answer these questions, we describe the development of FreeBu#3 (RQ1) in Section 6.3, the user study that investigated users' perceived values towards the tool (RQ2) in Section 6.4 and the analysis of users' interaction data (RQ3) in Section 6.5.

6.3 FreeBu#3 (RQ1)

FreeBu#3 has four visualizations/views to accommodate the users' comprehension of their online friends and help users create friend groups semi-automatically. The four visualizations/views are described as follows:

Circle View The circle view is the visualization we designed in FreeBu#2, as detailed in Section 4.3. This view corresponds to the inner-circle grouping strategy. Mutually connected friends tend to be put in the same circle. However, as discussed previously, people in the same school, family, workplace, etc. may also have connections with one another, the modularity-based communities could resemble the communities that are generated purely based on relevant attributes. In the circle view, each grey dot represents a friend. The dots align into different circles to represent communities. Zooming into a circle can divide this circle into sub-circles. The user can mouse-hover a dot to highlight it (Figure 6.1, top), or mouse-hover and right-click the central node of a circle (blue) to highlight all the members of the circle (Figure 6.1, bottom). The user may drag and drop a circle or an individual node to compose her own Facebook friend list (Figure 6.1). The group circles with different sizes also provide the

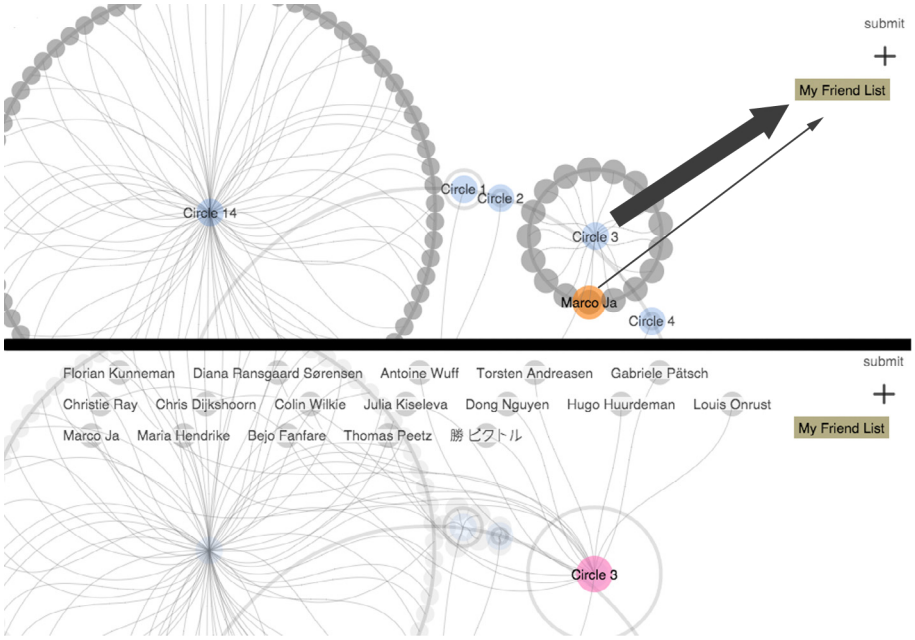

Figure 6.1: The circle view in FreeBu#3

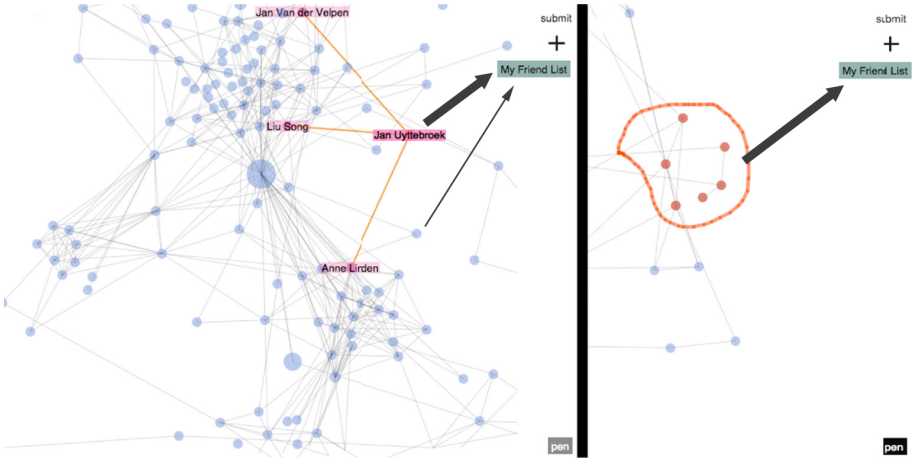

Figure 6.2: The map view in FreeBu#3

user with a sense of ordering, helping her quickly find outliers or surprising circles.

Map View The map view is newly designed to accommodate the inner-circle grouping strategy and the mutual-friend strategy as well. Different from the circle view, it directly shows the user's ego-network structure (i.e. friend graph) and provides various user interactions to manipulate parts of the network. The user's ego-network is directly visualized in a force-directed layout with the Fruchterman-Reingold algorithm [62], which is a typical graph-layout algorithm. It pulls connected nodes together and pushes disconnected nodes apart. Users can easily observe visual clusters and hub-nodes. The user can zoom and pan to explore the graph, zooming-in brings out the node labels. Mouse-hover on a friend node also brings out the friend's name label, meanwhile highlights the connections of this friend on the graph. Right-clicking a friend node will automatically select this node as well as its neighbours. User can then drag and drop the selected nodes (singular nodes or a node with its immediate neighbours) to compose her own friend list (Figure 6.1, left). Furthermore, the nodes' radii are set proportionally to the corresponding Betweenness [132] values so that the important nodes that connect different parts of the user's ego-network are enlarged and emphasized. It has been shown that the bridging structure in a user's ego-network is important for predicting strong social ties, such as romantic partners [7]. A node's Betweenness is equal to the number of shortest paths from all vertices to all others that pass through that node. A node with large Betweenness score indicates its central role in connecting different parts of the user's social network. We linearly map the nodes' Betweenness centrality values to their diameters ranging from 10 pixels to 30 pixels. To make group creation more flexible, the point-in-polygon function is implemented. The user can turn on this function by pressing the "pen" button on the bottom-right corner of the canvas and draw a polygon to enclose and select the nodes of interest, and drag-drop the selected nodes to compose lists (Figure 6.1, right).

Column View The column view is to accommodate the shared-attribute strategy. It generates the groups of friends based on common profile-attributes between friends, which is a generalization of Facebook smart lists. Each column represents a group. The "head" of the column is labeled with the corresponding attribute-value name. The "body" is a stack of friend name tags belonging to that column. If a column contains more than N_{col} (e.g. $N_{col} = 12$), only N_{col} friend tags are initially shown in the body of the column, with the "..." symbol to indicate there is more tags. Mouse-hover on the head of a column expands the column and show all the member names. The heights of the columns are proportional to corresponding the numbers of friends. Users can scroll left or right with mouse wheel to explore the columns. They can click the "overview" button for a summary of all the column labels. The user can drag and drop a column or an individual tag to compose lists (Figure 6.3). Moreover, the user can drag and drop columns into the "intersection" area at the bottom of the canvas. This area keeps the members that satisfy the attribute values from

Figure 6.3: The column view in FreeBu#3

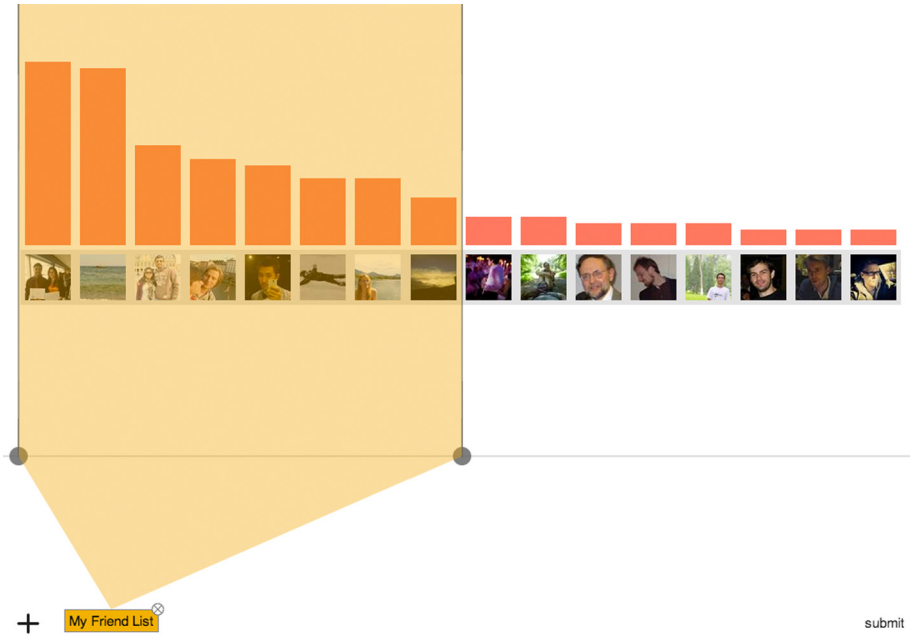

Figure 6.4: The rank view in FreeBu#3

the columns. The user can then use intersected area (also via drag-drop) to compose friend lists.

Rank View The rank view is to partially accommodate the contact-type strategy and the evaluative strategy. Studies [163, 50] have shown that interaction frequency linearly corresponds to the strength of interpersonal ties. However, this is rigid approximation. It does not cover the situation where offline close friends do not chat frequently online. It also does not addres the situations where the opinions of the ego-user towards his friends need to be considered to produce friend-grouping. In fact, accessing one's opinions towards his friends through the online ego-network data is a very complicated problem. While it may be relatively straightforward to infer closeness of relationship between two people via their online communication records, it is much more difficult, to infer opinions or emotions such as "respect", "dislike", or even "scornfulness", as there could be a large inconsistency between what a person thinks and what he expresses online. Also, people tend to suppress negative expressions in public [82]. Therefore, regarding the contact-type and the evaluative strategies, we also rely on the other views (circle, map and column) that help the user organize the relationships in his ego-network, and let the user make his own decisions.

We visualize the users' friends by aligning their profile photos horizontally near the middle of the canvas. The photos are ranked according to the communication frequencies of the user with his friends in Facebook chat. On top of each photo, a bar is shown if there is a communication history of the user with that friend. The more frequently the user chatted with a friend, the higher the bar is. The user can scroll left or right with mouse wheel to see the bars and photos. Mouse-hover can enlarge a photo can brings out the name beneath it. The user can select one or more friends by moving the two "knobs" with vertical lines. Clicking on a user-defined list "absorbs" the friends that are "clipped" by the two knobs into that list (Figure 6.4).

The four views share a similar way for creating customized friend lists. The user starts by clicking the "plus" button to add a new, empty list, aligned on the right (in the first three views) or the bottom (in the rank view) of the canvas. Each list is shown as a rectangle. The user can right-click a list to edit its name. Drag-drop actions put selected friends into a list, as illustrated in Figure 6.1, 6.2 and 6.3. In the rank view, "clipped" friends are put in a list by user clicking on the list, as shown in Figure 6.4. The user can submit the lists to his Facebook account by clicking the "submit" button. To help users learn to use the tool, we also created illustrations and instruction videos on the website. However, due to Facebook API change, the visualizations in Freebu#3 are no longer in service. We describe the new FreeBu that is independent of Facebook API in Chapter 10.

6.4 Perceived Values of FreeBu#3 (RQ2)

To investigate the perceived values of FreeBu#3, in the joint work with SMIT, VUB[2], we invited participants to explore the tool and fill out survey questionnaires[3].

6.4.1 Participants and Method

We contacted different intermediaries, mainly the heads of Flemish boy scouts, to disseminate an invitation letter to their colleagues and many young adults, via email or OSNs. The invitation letter included both URLs of FreeBu#3 and the corresponding survey questionnaire. We also advised potential participants in the letter to first check the video tutorials on the tool website. The button to open the tutorial page was immediately and distinctly visible on the homepage. The tutorial page contained detailed video instructions and illustrations that explain the purposes of the website and how to interact and make use of the visualizations. The participants were asked to test FreeBu#3 at home. No further instructions were given. We let the participants use the tool in a natural state. After usage/exploration of the tool, they then answered the questionnaire. By participating, people had the opportunity to win cinema tickets. In total, 49 participants explored their Facebook ego-network with FreeBu#3 and filled out the survey. The age range is 16-34 ($M = 21.6$, $SD = 5.0$), 20 males. These participants had 100-1935 friends on Facebook ($M = 465.3, SD = 327.1$).

The questions were grouped according to the usability, the perceived values of the tool and users' general perceptions towards the four visualizations (see Appendix B). The usability items were adapted from the Computer System Usability Questionnaire [114]. The "perceived-value" questions were an operationalization of the interviews and the coding process from our previous user study in Section 3.6. That is, these questions were informed by the previous interviewer's interaction with the interviewees. The previous user study identified three categories of perceived values for FreeBu#1. They were removing friends, grouping friends and reflection. There categories were translated into more detailed survey items in Appendix B. More specifically, the first three items can be considered a direct translation of the three categories (PV1–PV3), whereas the other items are further elaborations. All questions were measured on a 7-point Likert scale. When needed, we included the option 'I don't know' or

[2]The research group of Studies on Media, Information and Telecommunication, Vrije Universiteit Brussel

[3]The survey was conducted by Ralf De Wolf from SMIT, VUB.

'not applicable'. The negatively worded items are reversely scored, as indicated with "(-)" in Appendix B.

Of the 49 participants, 40 allowed us to collect their Facebook data and record their interactions with the tool, while nine chose to opt out of data collection. We refer to these 40 participants as user-set $\mathcal{A}$. In addition, 43 users who were not part of the user study also used the tool. These users are mainly university researchers and students who first heard of the tool and decided to use it, or came across it unintentionally. Out of these users, 39 users consented for data collection. According to their Facebook data, these users were similar to the participants: ages ranged from 17 to 36 (M = 23.9, SD = 5), 24 males, the number of friends ranged from 29 to 820 (M = 349.3, SD = 169.0). We refer to these 39 users as user-set $\mathcal{B}$. In Subsection 6.5, we report on interaction data with FreeBu#3 from both user-set $\mathcal{A}$ and $\mathcal{B}$.

6.4.2 Results and Interpretation

In Appendix B, the average score and the corresponding standard deviation are provided for each survey question. The results showed that every perceived value item is scored higher than the neutral 4 on a 7-point Likert scale. The participants especially valued FreeBu#3 for grouping and reflection. Wilcoxon Signed Rank tests [183] showed that friend-grouping was significantly more valued than friend-removing ($z = -2.820$, $p < .01$), and reflection was more significantly more valued than friend-removing ($z = -2.699$, $p < .01$). No significant difference was found between friend-grouping and reflection.

The scores on the usability items were also high. The highest average scores were among the items USAB 3, 5, 7, 8,10, which emphasized easiness of use (USAB 3, 8), clear and helpful instructions (USAB 5, 7) and practical friend-grouping functions (USAB 10). We also received critiques from 15 participants' free-form comments. Many indicated that not all visualizations were displayed or loaded very slowly, e.g. "zooming was tiresome and slow." This was mainly due to the performance bottleneck of browser-based data visualization, where animation was lagging for rendering thousands of visual objects.

The mean scores for all items regarding the visualizations were also generally higher than 4 on a 7-point Likert scale. The visualizations with less number of visual objects and more regular shapes were considered by the users better arranged and more pleasant to see, for example, the circle visualization and the rank visualization. The visualizations that correspond to the inner-circle strategy and the mutual-friend strategy, namely the circle and the map, gained relatively higher scores on whether the visualization "provided a clear image

of who my Facebook friends are". Comparatively, the column and the rank visualizations did not provide as good an overview on the user's friends.

Questions were also asked regarding each visualization's targeted effects, as listed below (with mean scores):

CIRC 4 - The circles could match with a grouping I would make. ($M = 4.62$)
CIRC 5 - The Facebook friends who were grouped together also belonged together. ($M = 4.57$)
MAP 4 - The map indicated how groups of friends are connected with each other. ($M = 5.14$)
MAP 5 - The map indicated which groups are completely segregated. ($M = 5.14$)
COL 3 - The characteristics in the columns were relevant. ($M = 4.10$)
RANK 5 - The ranking provided me with how much contact I have with my Facebook friends. ($M = 4.30$)

We can see that the map visualization, despite its relatively lower scores on clear spatial arrangement, gained high scores on showing connections and separations of friends, which were its intended effects. The circle and the rank visualizations gained moderate scores on their respective functions, which were "explicit grouping proposal" and "ordering friends based on chat frequency". The column visualization received relatively less points for its relevance of attribute-selection. This could be attributed to that (1) we did not take such profile information as "hobbies" and "family" into account, as we found it rare that people filled out such information in their Facebook profiles, (2) a large number of columns of friends based on education, workplace, etc. were generated, many were not of interest to the user.

6.5 User Interactions with FreeBu#3 (RQ3)

In this section, we report on the interaction data from the 40 survey participants (user-set $\mathcal{A}$) and the other 39 users (user-set $\mathcal{B}$).

6.5.1 Interaction Measures

In line with the exploratory nature of our study, we focused on descriptive statistics. Because eye movement is a well-known indicator of attention [72], and mouse positions highly correlate with eye-gaze positions [35, 96], we use #checks – the count of mouse clicks or enters on a visual object (circle, rectangle) – to measure user attention to the entity (i.e. Facebook friend) that the visual object represents.

Table 6.1: The number of users who used each of the four visualization, for the user-set $\mathcal{A}$, the user-set $\mathcal{B}$ and the union set

User-set	Circle	Map	Column	Rank
$\mathcal{A}$	30	29	26	31
$\mathcal{B}$	27	30	24	21
$\mathcal{A} \cap \mathcal{B}$	57	59	50	52

We complement this by measures that take the effect of the visual context of the object into account. In the circle visualization, we use the average percentage of friends checked by the user in a circle (%checked Friends) and the average number of checks per checked friend per circle (#checks/friend/circle) over all circles from all the users. The former is to indicate the extent to which the user interacts with a circle. The latter is to indicate the extent to which the user puts his attention on individual nodes in a circle. For example, a circle contains 5 friends, when the user checks 3 of them, the percentage of checked friends in this circle is 60%. When the total number of checks on this circle is 18, the number of checks per checked friend in this circle is 6.

In the map visualization, we use the average number of checks on an individual friend node (#checks/friend) to indicate the extent to which the user interacts with individual nodes. Furthermore, we use Mahalanobis distance [44] to quantify the relative position of a node in the map visualization. The node that is far away from the other nodes on the screen (i.e. the outliers) will be assigned a relatively high score, while the node that "belongs to the crowd" will be assigned a relatively low score. The scores are normalized for inter-group comparison.

In the column visualization, we use the average number of checks per column (#checks/column) over all the columns from all the users, to measure the extent to which a user interacts with a column.

In the rank visualization, similar to the map visualization, we use #checks/friend, only that each friend is represented as the friend's photo with bar chart.

6.5.2 Results and Interpretation

Table 6.1 shows the respective numbers of users in the four visualizations, from which we can see that the circle and map visualizations attract slightly more users than the other two. This is consistent with the survey scores on the

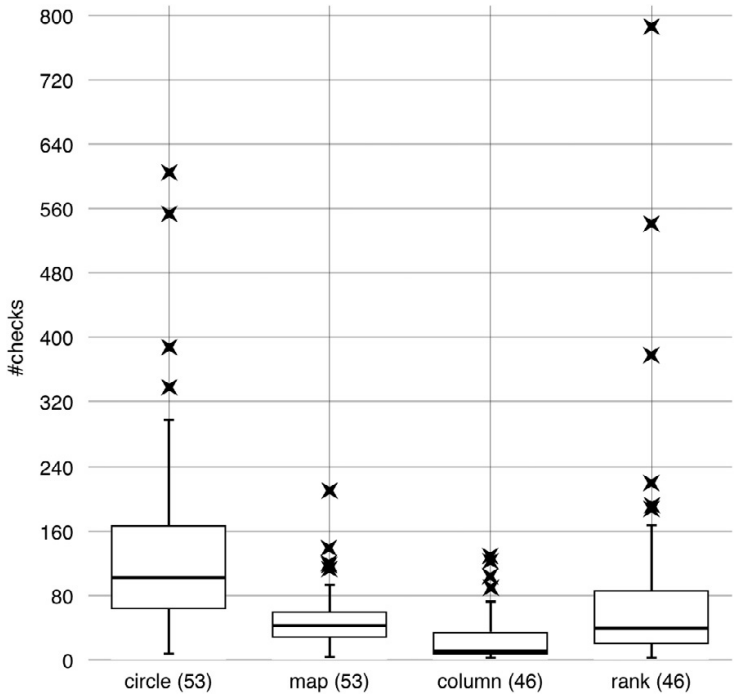

Figure 6.5: The boxplot of #checks for each visualization, the cross signs are the outliers, the numbers in the brackets indicate the counts of non-outliers.

respective targeted effects of the four visualizations. We can also see that the user-set $\mathcal{A}$ and $\mathcal{B}$ have similar user distributions over the four visualizations.

We then performed four two-tailed Mann–Whitney U-Tests ($\alpha = 0.05$) between $\mathcal{A}$ and $\mathcal{B}$ for the number of mouse checks in the four visualizations, and found that the difference between the two user sets in each visualization is insignificant. Therefore we merged the two sets of users, and summarize our explorative analysis results in Figure 6.5. From this figure we can see that the users in the circle visualization generally have the most mouse checks, while those of the column visualization have the least. The median #checks for the map and the rank visualizations are similar 43 and 40, respectively. But the rank visualization has a broader upper quartile, Whisker, and outlier ranges than those of the map visualization, and is comparable to those of the circle visualization. This indicates that the rank visualization also attracts more mouse checks compared to the map and the column visualizations. This is consistent with the survey scores on visual arrangement and pleasantness, which indicates that these two factors attract users' attention.

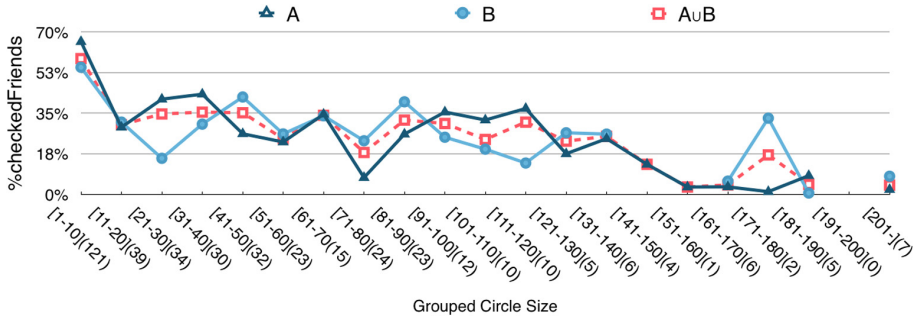

Figure 6.6: The percentage of checked friends, grouped by circle size

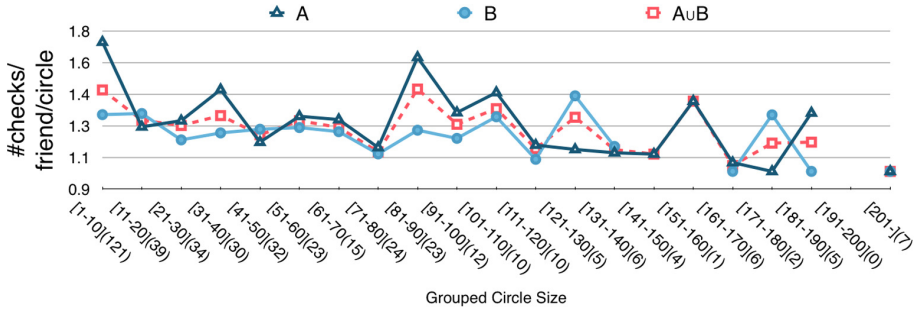

Figure 6.7: The number of checks per friend within a circle, grouped by circle size

Furthermore, we measured the session time that a user stays within the FreeBu, and found that the average session time per user is 13.8 min, with standard deviation 15.2 min (excluding an outlier that exceeds 10 h). The median time is 9.3 min. We did not record the session times for different visualizations of each user, but indeed a more-fine grained logging would provide us more insight into users' behavior with the FreeBu in future work.

Interactions with the circle visualization The circle visualization shows an initial set of circles as friend groups for the user to interact. Due to the nature of this type of visualization, the differences in the sizes of the circles may influence the user's subsequent interactions. We use the two metrics %checked Friends" and #checks/friend/circle to measure the extent to which a user interacts with a circle, as described in Subsection 6.5.1.

Figure 6.6 and 6.7 show our observations on these two metrics, with grouped circle size, for $\mathcal{A}$ users, $\mathcal{B}$ users and the union set. Each dot represents an averaged value corresponding to the group of circles with a size range (e.g.

[1–10]). The number in the brackets is the total number of circles with that size range. For example, [11–20] (39) indicates that there are 39 circles with size between 11 and 20 friends. From Figure 6.6 we can see that, when the circle size is not extreme, between 11 and 130 friends, the percentage of checked friends remains stable around 30% for the majority of the circles, regardless of their sizes. This pattern is similar in $\mathcal{A}$ and $\mathcal{B}$ users. This suggests a "checking-threshold" for a user to "comprehend" a group of visual objects. In other words, the user only needs to "check" a limited number of friends in a community circle to know "who are in the circle, and what this circle is about". This also supports previous evidence that modularity-based circles indeed make "sensible" circles for the user. Furthermore, from Figure 6.7 we can see that a friend node in a circle is typically checked 1.3 times by a user (for both $\mathcal{A}$ and $\mathcal{B}$) across the whole range of circle sizes.

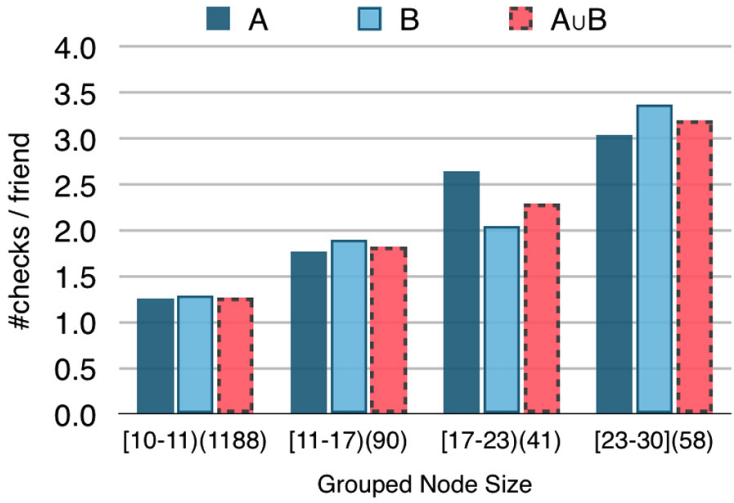

Figure 6.8: The average number of checks per friend node with grouped node size in the map visualization. To read the X-axis, e.g. [11–17) (90) indicates all the nodes with [11,17) pixel size, and there are 90 such nodes in total.

Interactions with the map visualization In the graph-layout of the map visualization, we look into how different sizes and positions of the nodes affect users' interactions with the visualization. It is expected to see that larger nodes attract more repeated checks from the users, as shown in Figure 6.8. This pattern is consistent in both user-set $\mathcal{A}$ and $\mathcal{B}$. Figure 6.9 groups the normalized Mahalanobis distances with a 0.2 interval. From the corresponding #checks/friend, we see that the nodes with large Mahalanobis scores attract more user interactions, which is especially the case for set $\mathcal{A}$ users. Recall that

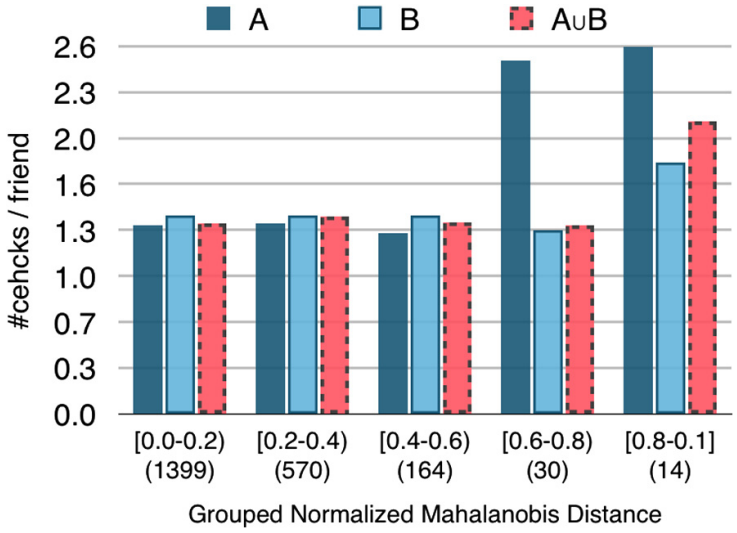

Figure 6.9: The average number of checks per friend node with grouped Mahalanobis distance in the map visualization. To read the X-axis, e.g. [0.2–0.4)(570) indicates all the nodes with their normalized Mahalanobis distance in [0.2, 0.4), and there are 570 such nodes in total.

the nodes with large Mahalanobis scores are the ones that isolated from the rest of the nodes, i.e. outliers. Figure 6.10 shows two example map visualizations, in which if a node is checked, it is colored. The more number of checks, the darker the color becomes. We can see that not only users focus on the larger nodes, but also focus on the "bridging" and "isolated" parts of a map visualization.

Interactions with the column visualization In the column visualization, large columns (the ones containing more friends) are placed to the left of the screen, so that the user will firstly see. However, we find that users seek to interact with specific columns in which they are interested, despite the initial positioning of the columns. As shown in Figure 6.11-top, large columns are age-related, while the other types of columns such "hometown", "edu" (education), "work" are much smaller. But from Figure 6.11-bottom, we see that the average numbers of checks per type of column by the users are concentrated on these smaller columns, especially education-related ones. We can also observe this pattern is similar between $\mathcal{A}$ and $\mathcal{B}$ users.

Interactions with the rank visualization In the rank visualization, the friends with whom the user has chatted most frequently online are put to the left of the screen. When visualization starts, the user sees the first 16 of these

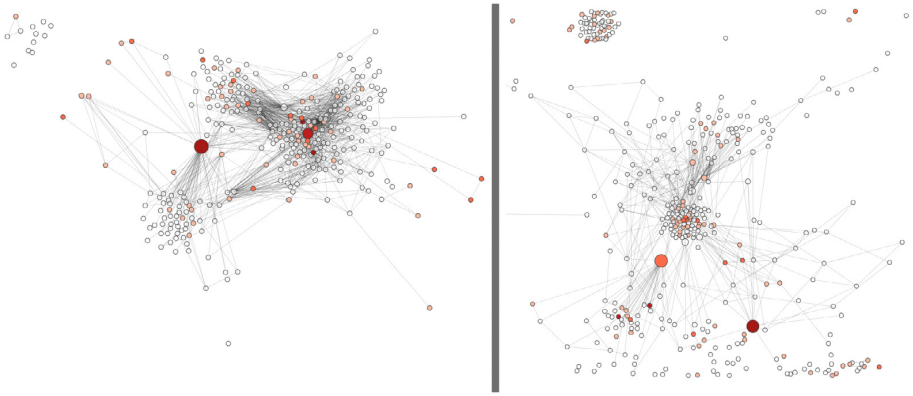

Figure 6.10: Two examples showing users' interaction focus in the map visualization

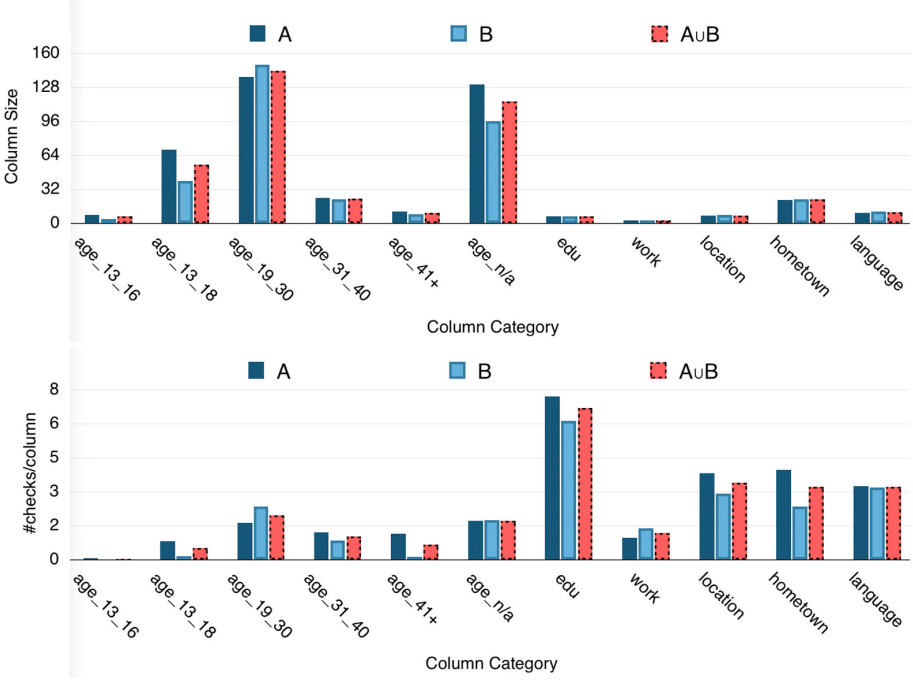

Figure 6.11: The column sizes (top) and #checks/column (bottom) on different types of columns in the column visualization

friends, as shown in Figure 6.12. Mouse scrolling to the right reveals more friends. We find that coincidentally, 16 is also the average number of friends with whom a user has chatted on Facebook in both user-set $\mathcal{A}$ and $\mathcal{B}$. Therefore, it is unclear which is the dominant factor that made users check the top 16 friends: (1) mouse scroll to the right to reveal the remaining friend photos for more interaction is costly, and users preferred to avoid it, or (2) users are more interested in interacting with the friend photos with histograms. It could be both factors that affected the users' mouse behavior.

Figure 6.12 shows users' ($\mathcal{A}$, $\mathcal{B}$ and the union set) average number of checks per friend (#checks/friend) on the top 20 friends, as they attract more than 90% of the total user interactions. From Figure 6.12, we can see that users checks the top two friends very frequently, then the number of checks starts to decrease, but increases again around the 15th friend. We can observe this pattern in both $\mathcal{A}$ and $\mathcal{B}$ users.

6.6 Conclusion

Comparing the different values items of all types of visualizations in the survey data it is not clear what the participant preferred. Only the column visualization received lower scores on providing a clear image of one's Facebook friends. The logging data showed how the participants paid attention to different aspects of their networked audiences. For example, often participants explored a similar percentage of a community of friends regardless of its size, or larger nodes positioned as bridges in the map visualization attracted users' attention. The rank visualization gives partially visualized individual friends, that is, the friends with whom the user has chatted. The results show how the participants mainly interacted with the top 20 of their Facebook friends, arguably their intimate friends. Finally, the logging data made clear that the column visualization received relatively less user-attention than the others. This could indicate that the profile-based friend groups presented in the form of columns of stacked names are less clear or interesting to the users. This finding is also supported by the survey results, considering the low score on providing a clear image of Facebook friends.

Our findings showed that users valued audience visualizations for reflection, next to grouping, and, when interacting with the visualizations, are drawn to different parts of their network. However, current OSNs do not offer an interactive and exploratory environment that allow the users to "play around" with their ego-network data, and reflect on the relationships they have online. We propose a shift from "audience control" to "audience transparency", and

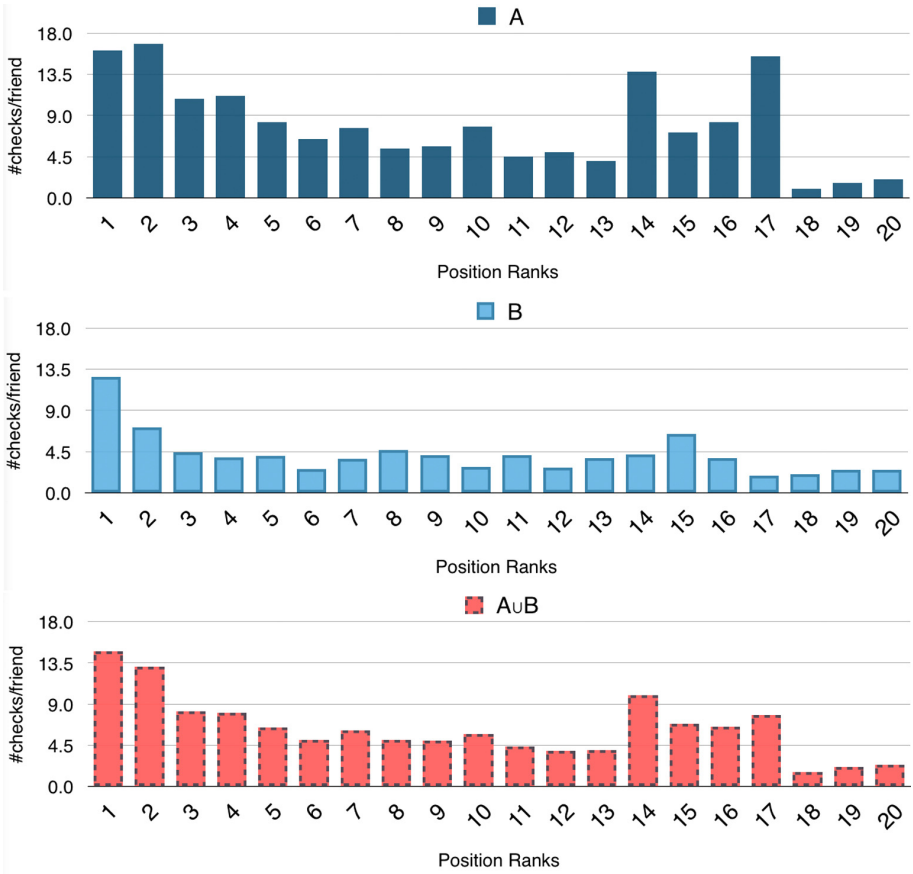

Figure 6.12: The average number of checks per friend on the top 20 friends in the rank visualization

move beyond an approach that is solely oriented on audience-control, but toward audience-understanding.

In terms of visualization design in FreeBu#3, we also identified a series of improvement points:

- In the four views, each friend is only represented by her name (the rank view also includes photos). More information, such as photo, profile, recent status and likes can be summarized in an "info box" that appears besides each focused friend.

- Search function needs to be implemented to accommodate the situation when the user wants to find a particular friend in the ego-network, either via that friend's name or certain properties of the friend.
- For the circle view, we notice that the user sometimes needs to zoom-in fairly deeply to reveal the friend names in a circle. This can be improved by adjusting the label-revelation threshold. Also, the positioning of the name labels needs adjustment, so as to avoid overlaps, while maintaining a grouping structure. Such adjustments are common in visualization design, which can be tackled in a more general approach, i.e. visualization library design that automatically resolves occlusion.
- The graph layout in the map visualization was considered less spatially orderly compared to the other visualizations. This is due to that (1) the optimization function in the force-directed layout does not directly correspond to good spatial arrangement perceived by human, (2) the network visualization with many interconnected nodes is inherently visually complex. We can improve this by introducing color coding and the features that organize the nodes in the same community into larger, singular nodes.
- FreeBu users have reported in some cases rendering visualizations is slow. Complex visualizations and user interactivity occupy a large part of browser resources, sometimes result slow response or crash. Though the current standard web technologies are encouragingly evolving, such as improved graphics rendering capabilities in HTML5, faster built-in Javascript engines, the browser-based computation power is still limited for large-scale, online, interactive visualizations. For tools like FreeBu, visualization programs need to be more economic.

Furthermore, while Wang et al. [178] indicated that the perception of one's audiences is dependent of their life stage, our user-study population is also limited to mostly adolescents and young adults users. Previous research has proven audiences visualizations to be useful for users of OSNs in making more aware decisions [115, 54, 123], using a task-based research design. We did not specify any particular tasks for the participants to focus on how they perceived and interacted with the technology in their natural state. This open approach allows for an alternative way to understand users' perceptions and behaviors, which are otherwise unobservable in controlled settings.

Part II

Aggregate Data Transparency

Chapter 7

Comparing Patterns in Social-sentiment Mining

In the first part of this thesis, we have investigated the online social privacy issue "context collision". We explored various solutions to this issue and developed a friend-grouping/exploration tool named FreeBu that helped Facebook users manage their contacts. However, thus far, we have been investigating solution approaches towards online social privacy, within egocentric networks. In other words, friend-grouping/exploration helps users mange their own online friends, on a micro-level. It is equally important, as privacy-as-practice (Section 1.2) and data literacy (Section 1.3) require, to help users gain insight into digitized and aggregated data on a macro-level. People's attributes and actions can be recorded. These records are digitized and aggregated by companies, organizations or individuals, subject to data analyzing, mining and repurposing. People should not only gain control of their ego-networks, but also have a good understanding of the data environment at large, of which they are part. One step towards this understanding is to promote aggregate data transparency through visualization tools and data mining techniques.

In this chapter, different from our previous work that focuses on egocentric networks, we look into the patterns of OSN data on a macro-level. More specifically, we investigate how Facebook users with different characteristics express sentiments, and how the texts posted on Facebook with different privacy settings exhibit different sentiments. On the one hand, we draw the link between our hypothesis testing and existing sociological research. On the other hand, we propose and implement algorithms that extract subgroup comparisons for the purpose of exploratory data analysis. We consider this work is of value to

researchers, OSN users and organizations who wish to gain insights on sentiment patterns in aggregated personal data from Facebook.

7.1 Related Work and Research Questions

Understanding users' sentiments in social media is important in many domains, such as marketing, sociological/psychological study and online application development. For example, in marketing, data analysts monitor and mine texts in social media to discover how participants in specific demographic groups react to certain brands or events. An analyst must be aware of existing sentiment differences. For instance, do older people express themselves more positively? Is there a difference in sentiment expression between married and single people? However, most hypotheses are based on offline studies. It is thus interesting to test and examine them in more detail with online social network data.

Recently, there has been a large interest in Facebook sentiment analysis [104, 158]. To the best of our knowledge, all the existing sentiment analysis has been conducted on status updates, or other (semi-)publicly available data in online social networks. While users post what they think or like publicly, they also chat privately[1]. Is there a sentiment difference between public and private? In this chapter, we discover and compare the sentiment patterns in both posts and chats on Facebook in a more differentiated way.

We use the term "sentiment" to refer to a simplified attitude or emotional state that can be characterised as positive, negative or neutral. Suppose that, for a given document, the degree of its positiveness is $s^+ > 0$ and the degree of its negativeness is $s^- < 0$, we consider the sentimental expressiveness of this document to be $(s^+ - s^-)$.

When a user posts or chats on Facebook, each post/chat has an audience range, mostly definable by the user. For example, a post's privacy setting can be adjusted from only visible to oneself or to the entire web. The number of participants also implicitly defines the audience range of a private chat. It is expected that people express more positive than negative sentiment on Facebook, as negative emotions are not socially favourable and people tend to suppress negative emotions in public [82]. Different levels of privacy settings may trigger different sentiment expressions. Our corresponding set of research question is:

RQ1: In Facebook chats and posts, (a) do Facebook users express more positive and/or less negative sentiment in public than in private? (b) Is there a difference in expressiveness?

[1]Facebook Chat `https://www.facebook.com/help/332952696782239/`

Regarding gender differences in emotional behaviour, research [37, 160] has indicated that there is no convincing evidence to support the widely stated "women are more emotional than men".[2] But we also see studies suggesting that women are indeed more emotionally expressive than men. For example, in a questionnaire study [190], the subjects (university students in Italy) were shown scenarios described in a vignette format. It was reported that women mention both positive emotions (such as joy, gladness) and negative emotions (such as anxiety, sadness) more often than men. Studies in social media data analysis uncover evidence in MySpace comments [168], Facebook status updates [56] and Yahoo! Answer texts [106] that partially support this finding: female users express more positive sentiment than male, but there is no obvious difference for negative comments. There has been no analysis on both Facebook posts (including shares, with comments) and users' chat records. Our corresponding set of research questions is:

RQ2: In Facebook chats and posts, (a) do women post more positive texts? (b) Is there a negative sentiment difference? (c) Are women more sentimentally expressive than men?

For age differentiated emotional behaviour, Gross et al. [81] investigated subjects' emotional experience, expression and control. The results consistently showed that, compared to younger subjects, older subjects reported fewer negative emotional experiences and greater emotional control. Older subjects are also reported to have lesser expressivity. However, do these findings translate to the communication in online social networks? Farnadi et al. [56] showed that, among 5,865 Facebook users, older people are more likely to express their positive and negative emotions in status updates, though the numbers of users in respective age groups were not reported. It was also mentioned that more than 40% of the young users were without emotions, though this could be attributed to the incomplete dictionary used. Furthermore, Stone et al. [162] found that people's positive emotional state increases after 50 years old. Stress and anger steeply declines from the early 20s, Worry was elevated through middle age (30-59 years old) and then declined. Our corresponding set of research questions is:

RQ3: In Facebook chats and posts, (a) are older people more sentimentally expressive? (b) Does negative sentiment decline after people's early 20s but increase during middle age? (c) Are people older than 50 years old more positive?

Researchers have also studied emotional differences in terms of relationship status. For instance, Yap et al. [188] found that married people (the number of subjects N=1,366, average married age: 29.8 years old, from the U.K.) were

[2]Subjects' facial expressions, speech, visual behaviour and/or self-report of feelings were analyzed for indications of emotionality.

happier than they would have been if they remained single. Another study [165] showed that being in a romantic relationship was associated with higher levels of anger (N=49, 18-20 year olds, from the U.S.). Our corresponding set of research questions is:

RQ4: In Facebook chats and posts, (a) do married people express more positive and/or less negative sentiment than single people? (b) Do people in a romantic relationship (not married) express less positive and/or more negative sentiment than single people? (c) Is there a difference between the people who are married and those that are in a relationship?

Note that, due to the features available in our dataset (see Section 7.2.4), we focus our investigation on the demographic attributes "gender", "age" and "relationship-status". Finally, most studies have focused on the correlations between singular (demographic) factors and sentiments in online social networks, it can be more insightful to study the sentiment differences using multiple factors. For example, the male users of 21-24 years old with the "friends" privacy setting (see Section 7.2) are less positive than those of 25-28 year old. This type of pattern mining falls under the Subgroup-Discovery paradigm [102, 186], which is to discover subgroups of multiple attributes in exploratory data analysis, without preconceived hypotheses. Our corresponding research question is:

RQ5: How can we discover "interesting" subgroup comparisons that help us gain more knowledge with multi-attribute groups?

7.2 Data

In this section, we first give an overview of the dataset (Subsection 7.2.1), then detail how we process the texts in multiple languages (Subsection 7.2.2) and describe the data attributes used throughout the chapter (Subsection 7.2.3, 7.2.4).

7.2.1 Data Collection and Overview

During November, 2013 - January, 2015, we collected the egocentric Facebook data of 199 FreeBu#3 users, with their consent. The box plot [170] for the number of friends of each user is shown in Figure 7.1. For each of the 199 ego users, we collected the profiles of the ego user and his/her friends. A profile[3] may include a user's name, birthday, home town, education, work experience,

[3]A full specification can be found at `https://developers.facebook.com/docs/graph-api/reference/v2.2/user`.

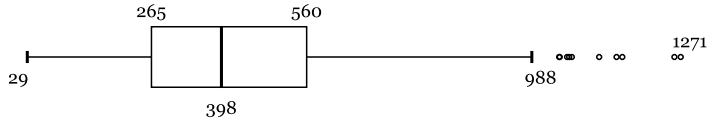

Figure 7.1: Box Plot for Each Ego User's Number Of Friends. The minimum, 1st quartile, median, 3rd quartile and maximum are 29, 265, 398, 560 and 988 respectively. The 10 outliers (of 199 ego users) are represented by circles.

.

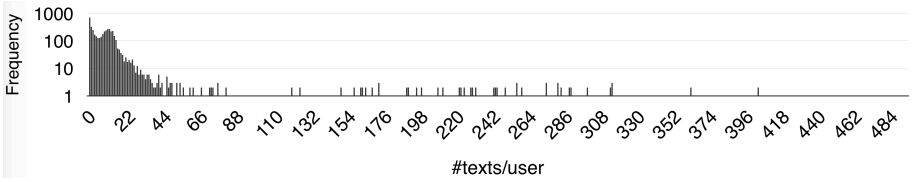

Figure 7.2: Histogram of #texts/user Frequencies in *chats*

.

spoken languages, etc. In total, we identify 66,013 such users with profiles, with 49.2% male, 50% female and unspecified for the rest. 64.6% of these users specify their birth dates, mostly people in their 20s. For more detailed information of gender and age groups, see Table 7.4 and 7.5 in Section 7.2.4. Furthermore, 61% specified their home towns, out of whom 68% come from Belgium, the rest mainly from Spain (5%), the Netherlands (3%), Germany (3%), Italy (2%) and France (1%).

Moreover, we collected users' Facebook chats[4]. In a user's chat history, some chatting participants may come from outside the user's ego-network, for chat messages can be exchanged between non-friends. We also collected the newsfeed posts (with comments)[5] from 42 ego users and their friends, in total 15,172 users. In both these *chats* and *posts*, each time a user types in a piece of text[6] to communicate. We use "text" for short. We use #texts/user to refer to the distribution of the number of texts sent by a user in a user set, #words/text and #chars/text to refer to the distributions of the number of words (space-separated) and the number of characters (UTF-8) in a text. Shapiro-Wilk tests [151] on both the original values and log-scaled values of #texts/user, #words/text, and #chars/text show that the respective distributions are

[4]We gave explicit notice that the chats would be collected with the user's consent.

[5]A full specification can be found at `https://developers.facebook.com/docs/graph-api/reference`.

[6]It is typically a short sentence or phrase, in the main message or the comment section of a chat or post.

Table 7.1: Data Summary of *chats* and *posts*

User Set	#users	#texts	#texts/user	#words/user	#chars/user
chats	4,480	84,751	10 (11)	6 (9)	23 (35)
posts	281,915	2,183,521	2 (3)	6 (9)	29 (41)

Table 7.2: Data Summary for Major Languages

	chats		*posts*	
Languages	#texts	#users	#texts	#users
Dutch (nl)	42,607	3,268	400,349	73,497
English (en)	10,977	1,894	635,997	117,521
Spanish (es)	1,835	347	247,358	42,672
German (de)	4,162	851	65,978	18,254
French (fr)	1,086	425	38,952	10,745
Italian (it)	867	377	180,211	32,415

significantly non-normal ($p < .001$). In fact, they are highly skewed. Indeed, we would expect exponential distributions here. Figure 7.2 shows an example. The median and IQR (Inter-Quartile Range) values (IQR values in brackets) of #texts/user, #words/text and #chars/text are summarized in Table 7.1. We also see that the texts that people typed in *chats* and *posts* are short.

7.2.2 Language Identification

In order to automatically detect the sentiments of the texts, we first need to sort the texts based on the languages in which they were written. However, language identification is non-trivial because of the corpus' large size, the many users from different countries and the short lengths of the texts. It is known that language identification becomes more difficult as the number of languages increases, and the length of the document decreases [10]. Lui and Baldwin [121] selected and compared eight language identification systems on labeled Twitter texts. They showed that an equal-weight voting over three systems consistently outperforms any individual system. These systems are: LangID [118], LangDetect [131] and CLD2.[7] We adopt this method to identify the languages of the sentences in the corpus. HTML tags and URLs have been removed beforehand from the corpus. The results are summarized in Table 7.2.

[7] `https://code.google.com/p/cld2/`

Table 7.3: Data Summary for Privacy Levels

chats				*posts*	
#participants	#texts	#users	Privacy Setting	#texts	#users
2	53,983	3,249	*public*	362,038	71,665
[3, 4]	4,054	572	*FoF*	67,151	13,737
[5, 6]	1,625	393	*friends*	1,147,141	177,350
[7, 10]	1,698	529	*custom*	144,990	28,860
[11, 20]	1,130	480	*self*	17	6
[21, 64]	329	235			

In total, 70,389 texts in 48 languages (83.1% of the original texts) from *chats*, and 1,890,476 texts in 66 languages (86.6% of the original texts) from *posts*, have been identified respectively. The languages of most texts in both *chats* and *posts* are Dutch, English, Spanish, German, French and Italian, as shown in Table 7.2. The unidentified texts are usually very short phrases that are abbreviations, internet slang, (intentional) typos, emoticons and exclamation marks, such as "-_-||", ":)", "STUDYYYYY!!!", "lmao!", or that occur in multiple languages such as "hehe","amen". Note that not all texts with identified languages can be processed by the sentiment analysis tool (see Section 7.3) due to the limited language packages available. Eventually, we analyzed 74.1% *chats* and 78.7% *posts* in 11 languages, which we take as the discourse.

7.2.3 Privacy Levels

Each newsfeed post or chat record, with its comments, has an audience range, namely the set of (Facebook) users who can see the text. The texts in a chat are only visible to the chat participants. We can differentiate levels of privacy by the number of participants in a chat. Most chats are exchanged between two people. The visibility of a text in a post is defined by its privacy setting, which has five levels: *public* refers to everyone on or off Facebook; *friends of friends* (*FoF*) refers to the friends of the author of the post, and the friends of these friends; *friends* refers to the friends of the author of the post; *custom* refers to a customised audience, for example, the user specifies a friend list, or individual friends who can(not) see the post; and *self* refers to only the author him/herself as the audience of the post[8]. The data statistics are summarized in Table 7.3. Most people set their privacy settings for posts to *friends*. The *public* setting is also prevalent, but this could be attributed to the legacy default privacy setting

[8]However, the posts with the *self* privacy setting can still be retrieved by third-party applications that have the user's consent.

Table 7.4: Data Summary for Gender

	chats		*posts*	
Gender	#texts	#users	#texts	#users
male	32,305	1,955	591,998	8,790
female	29,365	1,990	435,639	8,349

Table 7.5: Data Summary for Age

	chats		*posts*	
Age	#texts	#users	#texts	#users
[13, 16]	544	59	1,684	94
[17, 20]	15,516	754	65,676	1,776
[21, 24]	15,627	816	236,128	4,167
[25, 28]	7,696	480	167,839	2,454
[29, 32]	2,744	208	79,763	984
[33, 36]	1,405	99	36,088	399
[37, 40]	274	26	13,235	174
[41, 50]	419	54	24,629	303
[51, 60]	175	30	9,484	144
[61, 80]	129	8	989	37

on Facebook.[9] We can also see that there are quite some users making efforts to customize their privacy settings.

7.2.4 Profile Features

The data summary for the "gender", "age", and "relationship status" are shown in Table 7.4, 7.5, 7.6 respectively. We chose the age groups similar to [162]. Since it is unlikely for people older than 80 years old to use Facebook, we assume that these data are untrustworthy and exclude the corresponding users from our analysis for age, namely 5 users in *chats* and 27 users in *posts*. For "relationship status", we only consider *married*, *in a relationship* and *single*. Also, we find that 99.9% of the users do not specify their "religion", "political-view" and "interested-in" features[10] that are available in Facebook profiles, so we do not analyze these features further.

[9] `https://newsroom.fb.com/news/2014/05/making-it-easier-to-share-with-who-you-want/`

[10] "Interested-in" is a feature where the user can indicate whether he/she is interested in male, female or both in a potential, romantic relationship.

Table 7.6: Data Summary for Relationship Status

Relation. Status	*chats* #texts	*chats* #users	*posts* #texts	*posts* #users
married	626	36	81,029	856
in a relationship	4,673	158	205,004	2,462
single	2,818	190	196,107	2,089

7.3 Sentiment Analysis with SentiStrength

We need sentiment analysis tool(s) to detect the sentiments in the texts. SentiStrength [167] is a lexicon-based system that detects polarized sentiments (with strength) of short informal texts. It takes into account both terms and other language features such as booster words, negation, emoticons, etc. Thelwall et al. [167] show that SentiStrength outperforms other standard machine-learning algorithms such as SVM, J48 tree, Naive Bayes, etc. Abbasi et al. [1] further show that SentiStrength is generally better than the other 14 similar, stand-alone tools on five benchmark datasets.

To understand SentiStrength's performance in more detail, we look at several key datasets on which SentiStrength has been tested and the corresponding results. In [167], SentiStrength was tested on 1041 MySpace comments in English. Each comment contains 20 words and 101 characters on average. The positive/negative sentiment strength scores for a comment are $\{1, 2, 3, 4, 5\}/\{-1, -2, -3, -4, -5\}$, with $1/-1$ meaning no positive/ negative emotion or energy, and $5/-5$ meaning very positive/negative emotion or energy. Note that a sentence is given both positive and negative scores by SentiStrength.

Three human coders' scores on the same comments were averaged and taken as the ground truth for subsequent training and testing. Note that the inter-coder agreement is moderate, with pairwise overlaps ranging from 51.0% to 68.2% for both positive and negative scores. However, the relaxed overlap rates (within± 1) are over 94%. This indicates that people give different positive/negative scores for the same comment, but these scores often vary within the small ± 1 range. SentiStrength was later improved and tested [166] on more online user texts, including BBC forum posts, Digg.com posts, Runners World posts, Twitter posts and Youtube comments, with human-coded scores to some extent. It was shown to be better than standard machine learning methods on short, informal texts. Thelwall et al. [167] also showed that supervised (with term-weight adjustment) and unsupervised (with default term weigths) SentiStrength algorithms perform similarly, suggesting that it is robust on unlabelled data.

Table 7.7: Summary of Sentiment Strength Scores

	Chats		Posts	
	Positive	Negative	Positive	Negative
1	43,305 (68.9%)	55,552 (88.4%)	1,074,092 (62.3%)	1,598,748 (92.7%)
2	18,317 (29.2%)	6,664 (10.6%)	600,225 (34.8%)	102,543 (6.0%)
3	1,117 (1.8%)	359 (0.57%)	44,910 (2.6%)	17,642 (1.0%)
4	96 (0.15%)	263 (0.42%)	5,022 (0.29%)	5,767 (0.33%)
5	7 (0.01%)	4 (0.006%)	572 (0.03%)	121 (0.007%)

The within±1 accuracy of unsupervised SentiStrength is consistently higher than 90% (e.g. 97.8% resp. 95.6% on MySpace comments and 94.2% resp. 93.4% on Twitter data for positive resp. negative sentiment).

Originally developed for English, SentiStrength was then extended to accommodate other languages. We use the available language packages at hand: Dutch, French, German, Italian, Polish, Portuguese, Russian, Spanish, Swedish and Turkish. Among these languages, Dutch, German, Russian, Spanish and Turkish have been validated to some extent[11]. However, to the best of our knowledge, there is no direct evaluation of SentiStrength of alternative languages on human-coded datasets that have {±1,±2,±3,±4,±5} scores. The Spanish [71] version was developed and applied to analyze political party alignment on Twitter datasets. The predicted 5-level sentiment scores were converted to other levels (e.g. positive/negative or neutral) and evaluated. The results are moderate. For example, the accuracies for positive and negative sentiment are 0.68 and 0.56 respectively. As the levels of sentiment have changed, there were no within±1 accuracies. Similarly, in the Turkish [176] version, SentiStrength was evaluated on movie review datasets with positive or negative sentiment. The corresponding accuracies were shown to be larger than 0.7. The German version [126], is close to the English SentiStrength in terms of human-coding. The ground truth dataset is composed of 500 short texts, which were selected from Facebook, Youtube, Amazon comments, etc. Three native German speakers annotated the texts with 0 to ±3 sentiment scores. The pairwise inter-coder overlaps range from 66% to 76%. The predictive accuracies are 49.0% resp. 58.2% for positive resp. negative sentiment, but the within±1 accuracies were not reported.

We then test SentiStrength on the human-coded datasets in Russian (3,976 tweets) and Spanish (1600 tweets) that are provided on the website (see footnote 11). For the Russian version, the within±1 accuracy is 93.7% resp. 90.0% for

[11] according to `http://sentistrength.wlv.ac.uk/#Non-English`.

positive resp. negative sentiment. For the Spanish version, the within±1 accuracy is 90.0% resp. 90.3% for positive resp. negative sentiment. We use the default settings of SentiStrength in different languages. We assume that the positive and the negative scores of a text are generated independently and that the positive/negative scores of different texts are generated independently. We run SentiStrength on the texts in *chats* and *posts*. The counts of texts with positive and negative sentiment are summarized in Table 7.7. We see that most texts in *chats* and *posts* are without sentiment, especially for the negative sentiment.

7.4 Single-Attribute Hypothesis Testing

In this section we test the sentiment differences according to *RQ1–RQ4*. Because the sentiment scores are highly skewed (as shown in Table 7.7), we use the non-parametric Mann-Whitney U test [120] for two independent groups, and Kruskal Wallis test [105] for > 2 independent groups to test sentiment difference between attribute-groups. We report significant results[12] with two-tailed $p < .01$. The results in pairwise, post-hoc tests are Bonferroni-corrected [52]. We exclude the texts with unspecified gender, age or relationship status, as well as the *self* group in the privacy levels of *posts* (Table 7.3), as it contains few texts. Furthermore, we consider the merged [37,50] and [51,80] age groups (Table 7.5) in *chats* to account for larger group size. We test both positive ($s^+ \in [1, 5]$) and negative ($s^- \in [-5, -1]$) sentiment differences. Also, when needed, we test sentiment expressiveness ($s^+ - s^-$) differences. We will use the notation $\boldsymbol{G}_A \Bbbk \boldsymbol{G}_B$ with $\Bbbk \in \{>, <, \approx\}$ to denote group $\boldsymbol{G}_A$ is more, less than or similar to $\boldsymbol{G}_B$ in terms of the absolute values of positive sentiment, negative sentiment or expressiveness. Note that $\boldsymbol{G}_A > \boldsymbol{G}_B$ and $\boldsymbol{G}_B \approx \boldsymbol{G}_C$ does not automatically imply that $\boldsymbol{G}_A > \boldsymbol{G}_C$.

7.4.1 Sentiment Differences between Privacy Levels (RQ1)

Tests show that the private chats and public posts differ significantly in positive sentiment ($U = 5.1 \times 10^{10}$), negative sentiment ($U = 5.2 \times 10^{10}$) and expressiveness ($U = 5.2 \times 10^{10}$). More specifically, the texts in *posts* are more positive and expressive than those in *chats*. The texts in *chats* are more negative than those in *posts*. This indicates that people tend to express more positive sentiment in *posts* shared with a broad audience, whereas they feel

[12] We summarize and selectively report the results of the post-hoc pairwise tests in Subsections 7.4.1, 7.4.3 and 7.4.4. A complete report can be found at `http://goo.gl/R5k5iQ`.

Table 7.8: Age Group Expressiveness in Chats

$[13,16] > [21,24] > [25,28],[29,32],[37,50],[51,80]$
$[17,20] > [33,36] > [21,24] > [25,28],[29,32]$

more free to express less positive, and also less extreme sentiments in *chats* that are exchanged within a private circle of participants. This partially confirms our hypothesis in *RQ1* that there is indeed a general pattern that people are more positive and less negative in public than in private on Facebook. Within *chats*, there is a difference between the groups of different privacy levels in positive sentiment ($\chi^2(5) = 29.0$), and negative sentiment ($\chi^2(5) = 83.5$). The conversations involving [11-20] participants are both more positive and negative than those involving 2 participants. It coincides with the general pattern that the texts are more sentimentally expressive in a more public setting. In *posts*, there is a difference in positive ($\chi^2(3) = 840.6$) and negative ($\chi^2(3) = 130.1$) sentiments between privacy levels: the *FoF* (friends of friends) texts are both more positive and negative than the texts with other settings. The texts with the *friends* and *custom* settings are more positive than the *public* texts. This provides us with a more detailed insight: the texts with the"fairly public" *FoF* setting are both more positive and negative than others, but the sentiments of the texts with a complete public setting are generally reserved.

7.4.2 Sentiment Differences between Genders (RQ2)

There is a difference between males and females in positive sentiment ($U_{chats} = 4.1 \times 10^8$, $U_{posts} = 1.2 \times 10^{11}$) and negative sentiment ($U_{chats} = 4.6 \times 10^8$, $U_{posts} = 1.3 \times 10^{11}$). More specifically, females are more positive than males in both *chats* and *posts*. The females are also more negative in *chats* than the males. There is no significant difference regarding negative sentiment between males and females in *posts*. This shows that females are indeed more sentimentally expressive than males in general. While female users are more positive than male users, they can also be more negative.

7.4.3 Sentiment Differences between Ages (RQ3)

In *chats* and *posts*, there is a difference between age groups in positive sentiment ($\chi^2_{chats}(7) = 151.7$, $\chi^2_{posts}(9) = 4{,}998.3$), negative sentiment ($\chi^2_{chats}(7) = 123.1$, $\chi^2_{posts}(9) = 109.7$) and expressiveness ($\chi^2_{chats}(7) = 99.6$, $\chi^2_{posts}(9) = 4{,}403.0$). Post-hoc analysis reveals that younger people are generally more sentimentally

expressive. In *posts*, younger groups are always more expressive than older ones, except that there is no significant difference between [13,16] and [17,20]. We see similar patterns in *chats*, with the exception of the age group [33,36], as shown in Table 7.8. These findings generally contrast the hypothesis in *RQ3* that older people are more sentimentally expressive. We also find that the [17,20] group is more negative than older age groups in *posts*, which supports the hypothesis in *RQ3* that negative sentiment declines after the early 20s, but we do not see an increase of negative sentiment in the mid-age range ([33,59]). Interestingly, we see the opposite in *chats*. The [17,20] group is less negative than the people between 21 and 50 years old. The late teen [17,20] group seems to behave differently from older people in terms of negative sentiment expression. Furthermore, we find that younger people are generally more positive, including that the [51,80] age group is less positive than the other younger groups, which does not support the hypothesis that > 50 year olds are more positive.

7.4.4 Sentiment Differences between Relationships (RQ4)

Tests show that there is also a difference between relationship-status groups, in positive sentiment ($\chi^2_{chats}(2) = 66.7$, $\chi^2_{posts}(2) = 2{,}642.4$) and negative sentiment ($\chi^2_{chats}(2) = 66.7$, $\chi^2_{posts}(2) = 303.0$). More specifically, in both *chats* and *posts*, we find that the texts from *single* users express more positive sentiment than those from *married* users. There is no significant difference regarding negative sentiment between the two groups in *chats*, but the texts from *single* users express more negative sentiment than those from *married* users in *posts*. This finding shows a contrast with *RQ4(a)* in that *single* users express themselves more positively than *married* users. For *RQ4(b)*, in *chats*, we find that there is no significant difference between *single* users and the users *in a relationship* in positive sentiment, but the texts from *single* users are less negative than those from users *in a relationship*. This supports our hypothesis. In *posts*, we find a stronger confirmation that the users *in a relationship* express both less positive and more negative sentiment than the *single* users. For *RQ4(c)*, users *in a relationship* have more positive chats and posts than those from *married* users. The users *in a relationship* also have more negative posts than the *married* users. These findings also show that *married* users are more neutral than the rest in terms of expressing sentiment online.

7.5 Multi-Attribute Comparison Extraction (*RQ5*)

So far, we have analyzed the interplay between the groups of users defined by singular attribute values and corresponding sentiments. It is relatively

straightforward to apply statistical tests in such scenarios. However, we often need to look into the "behaviour" of user groups with combined attributes. For example, we find that the users with *married* relationship status tend to be less positive than the users with other statuses, but does this hold for both genders, different ages and so on? Recent advances in the field of subgroup discovery enable fast discovery of "quality" subgroups with high diversity and low redundancy [173], or discover subgroups with multiple numerical target attributes [113]. To the best of our knowledge, existing approaches extract subgroups that have unusual or distinct distributional characteristics with respect to the entire population. For example, the target values in the subgroup "the 25-28 year-old males" are compared with the entire population. If this comparison produces a high score according to a certain quality measure, it is considered an interestingly distinct subgroup. However, instead of individual subgroups, we are interested in "comparisons" between subgroups, such as "the 25-28 year-old males" versus "the 29-32 year-old males", or "the 25-28 year-old males" versus "the males with the age interval other than 25-28".

Furthermore, various quality measures are adopted or proposed to evaluate subgroups, and sometimes to prune the search space. But these measures often have a normality (Gaussian distribution) assumption for real-value target attributes (e.g. Mean Test, Numeric Weighted Relative Accuracy), whereas we see in Section 7.3, data could be non-normally distributed. We develop two top-down heuristic search algorithms, with statistical tests, without the normality assumption[13], as both quality measures and pruning strategy, to extract subgroup comparisons. These comparisons reveal interesting attribute combinations that provide a more fine-grained insight into the relationships between attributes and sentiments. They also offer potential sociological hypotheses for future study.

7.5.1 Vertical Comparison

Consider a hierarchy A of attribute types (labeled a_i) and values (labeled $a_{i,j}$), $i, j \in \mathbb{N}$, as shown in Figure 7.3. Namely, $A = \{(a_i, A_i)\}$, $A_i = \{a_{i,j}\}$. We define the complement of an attribute value $a_{i,k}$ within A_i as $A'_{i,(k)} = \{a_{i,j} | j \neq k, a_{i,j} \in A_i\}$. Similarly, the complement of an attribute a_i in the scope of A is defined as $A'_{(i)} = \{A_j | i \neq j, (a_j, A_j) \in A\}$. Note that the algorithms (Section 7.5.1 and 7.5.2) can be straightforwardly extended to accommodate a hierarchy with more levels of attribute values. For example, the age interval can be coarse initially, but divided into finer intervals at deeper levels. Consider a subgroup $\boldsymbol{G}$ as a set of attribute values ($a_{i,j}$) where each corresponding attribute type (a_i)

[13] because of the usage of Mann-Whitney U test

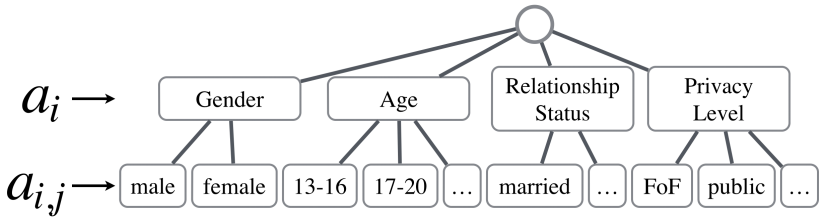

Figure 7.3: Illustration of Hierarchy of Attribute Types and Values

appears zero or one time. For example, a subgroup can be the females within 21-24 years old, namely {female, 21-24}. Let $\mathcal{G} = \{\boldsymbol{G}_i\}$ be the set of these subgroups. We use $\mathbb{m} \in M$ with $M = \{pos, neg, express\}$ to denote a chosen measure of positive sentiment, negative sentiment and expressiveness. We use the sign $\mathbb{s} \in \{>, <, \approx\}$, as defined in Section 7.4, to describe the relationship between two subgroups, with the measure $\mathbb{m} \in M$, according to the statistical test $\mathbb{t}$ and the significance level α. Let $\mathbb{t}(\boldsymbol{G}_A, \boldsymbol{G}_B, \mathbb{m})$ be the test that returns the sign $\mathbb{s}$ and the two-tailed p-value p, on subgroups $\boldsymbol{G}_A$ and $\boldsymbol{G}_B$ with the measure $\mathbb{m}$.

The algorithm for finding "vertical comparisons" of subgroups is detailed below. We use the set of "base comparisons" $\mathcal{C}_{base}$ to store the comparisons between an attribute value and its complement within the same attribute, namely $\mathcal{C}_{base} = \{(\{a_{i,k}\}, A_{i,(k)}, \mathbb{s}, \mathrm{m}, p)\}$, and the set of comparisons $\mathcal{C}$ to store the comparisons between a target subgroup $\boldsymbol{G}$ (with $|\boldsymbol{G}| > 1$) and its "counter part" S (with $|\boldsymbol{G}| = |\boldsymbol{G} \cup S| + 1$). We then have $\mathcal{C} = \{(\boldsymbol{G}, S, \mathbb{s}, \mathbb{m}, p)\}$. A depth-first search continuously looks for comparisons of subgroups with a larger number of attribute-value combinations. The significance level α serves as the pruning threshold that stops the search at a branch if the corresponding test p-value is larger than α (Line 19-22).

```
1: Given A, 𝕞
2: C_base ← ∅, C ← ∅, G ← ∅
3: for a_i do
4:     for a_{i,k} do
5:         𝕤, p ← 𝕥({a_{i,k}}, A_{i,(k)}, 𝕞)
6:         C_base ← C_base ∪ {({a_{i,k}}, A_{i,(k)}, 𝕤, 𝕞, p)}
7:         G_i ← {a_{i,k}}
8:         CompareInDepth(G_i)
9:     end for
10: end for
11: function CompareInDepth(G_i)
12:     if G_i ∉ G then
```

Table 7.9: Examples of Vertical Comparison

data, $\mathbb{m}$	comparison	base
chats, pos	{female, 21-24}<{female, ¬(21-24)}	{21-24} ≈ {¬(21-24)}
posts, pos	{17-20, relation.}<{17-20, ¬(relation.)}	{relation.}>{¬(relation.)}
posts, neg	{25-28, friends}>{25-28, ¬(friends)}	{friends}<{¬(friends)}

```
        G ← G ∪ {G_i}
        for A_u ∈ A'_(i) do
            for a_{u,k} ∈ A_u do
                G_u ← G_i ∪ {a_{u,k}}
                S_u ← G_i ∪ A'_{u,(k)}
                𝕤, p ← 𝕥(G_u, S_u, 𝕞)
                if p ≤ α then
                    C ← C ∪ {(G_u, S_u, 𝕤, 𝕞, p)}
                    CompareInDepth(G_u)
                end if
            end for
        end for
    end if
end function
```

The algorithm returns two filled sets of comparisons $\mathcal{C}_{base}$ and $\mathcal{C}$. $\mathcal{C}_{base}$ contains the comparisons of single attribute values and their complements, informing us whether and how an attribute value is distinguished from the rest. $\mathcal{C}$ contains the comparisons of attribute combinations in different orders and their more general counterparts, informing us that by adding a specific attribute value, whether and how a combination is distinguishable from the rest. Table 7.9 shows examples of vertical comparison with the smallest p-values. For example, while there is no positive sentiment difference between the chats from the people of 21-24 years old and other ages, adding the "gender=female" attribute value reveals that, in contrast, females of 21-24 years old have less positive chats compared to other females. We can also see that (3rd example) while the posts with the *friends* setting are generally less negative than those with other settings, the people of 25-28 years old express themselves more negatively in this setting.

7.5.2 Horizontal Comparison

While the vertical comparison helps us see the effect of adding/removing one attribute value on sentiment distributions, it is also desirable to see how different values of the same attribute affect sentiment distributions under more complex conditions. For example, how do {male, relation.}, {male, married}, {male, single} differ from each other? To this end, we modify the algorithm in Section 7.5.1 to extract horizontal comparisons, as shown below. Statistical tests are performed on a set of subgroups corresponding to all the attribute values $a_{u,k}$ under an attribute a_u, conditioned on a previously given subgroup $\boldsymbol{G}_i$ (Line 15-23). Furthermore, let $\mathcal{G}'$ ($|\mathcal{G}'| \geq 2$) be a set of subgroups subject to post-hoc analysis, and $\mathbb{t}(\mathcal{G}', \mathbb{m}, \alpha)$ the function that performs the pairwise testing[14] and returns a set of comparisons that are significant at α level. The notation is same as in the algorithm in Section 7.5.1, $\mathbb{m} \in M$ with $M = \{pos, neg, express\}$ denotes a chosen measure of positive sentiment, negative sentiment and expressiveness, and α serves as a threshold to remove the comparisons with large p-values.

```
1: Given A, m
2: C_base ← ∅, C ← ∅, G ← ∅
3: for a_i do
4:     G' ← ∅
5:     for a_{i,k} do
6:         G_i ← a_{i,k}
7:         CompareInBreadth(G_i)
8:         G' ← G' ∪ {G_i}
9:     end for
10:    C_base ← C_base ∪ t(G', m, α)
11: end for
12: function CompareInBreadth(G_i)
13:    if G_i ∉ G then
14:        G ← G ∪ {G_i}
15:        for A_u ∈ A'_(i) do
16:            G' ← ∅
17:            for a_{u,k} ∈ A_u do
18:                G_u ← G_i ∪ {a_{u,k}}
19:                CompareInBreadth(G_u)
20:                G' ← G' ∪ {G_u}
21:            end for
22:            C ← C ∪ t(G', m, α)
23:        end for
```

[14] Each pairwise test is Bonferroni-corrected in itself. However, we do not impose a global correction because of the heuristic data mining approach that we take.

Table 7.10: Examples of Horizontal Comparison

data, m	comparison	base
posts, express	{custom, 17-20}<{custom, 37-40}	{17-20}>{37-40}
posts, pos	{male, married}>{male, single}	{married}<{single}
posts, pos	{male, relation.}>{male, single}	{relation.}<{single}
posts, pos	{male, married}>{male, relation.}	{married}<{relation.}

24: **end if**
25: **end function**

Table 7.10 shows examples of horizontal comparison with the smallest p-values. For example, from Section 7.4.3 we know that younger people are more sentimentally expressive, as one base comparison {17-20}>{37-40} shows. However, when the privacy setting is *custom*, the expressiveness reverses, suggesting that the {17-20} group is not as expressive as they would be in a more public setting, and/or the {37-40} group express themselves more freely in a more private setting. Moreover, from Section 7.4.4 we know that in *posts*, the positive sentiment differences in relationship status are: {single}>{relation.}>{married}, but this pattern has reversed when adding the "gender=male" attribute value, as shown in the table, providing us with a more differentiated view on the positive sentiment differences in relationship status.

Note that in both the vertical and horizontal comparisons, the contrasts between a "deeper comparison" (e.g. {male, married}>{male, single}) and a base comparison (e.g. {married}<{single}) can be trivially extracted, we do not detail the corresponding algorithms here.

7.6 Limitations and Outlook

We apply statistical tests to identify differences between groups of sentiment scores, based on the assumption that each text's sentiment is independent of other texts' sentiments. This assumption has two limitations: first, the sentiments of the texts from the same user may be correlated; second, the sentiments of the texts from the same chat or post may as well be correlated.

Also, as seen in Section 7.2, the user sample in our dataset is biased. It consists of mostly young people from west European countries, particularly so for the users in *chats*, who are mostly Flemish students. Moreover, we only considered

the users who have available profile features for demographical factors, which increases the bias.

Furthermore, we used a tool (SentiStrength) to extract sentiment scores from the texts in multiple languages, which is bound to produce errors. Although it has been shown to be encouragingly accurate in relevant domains (Section 7.1, 7.2), it is yet to be investigated to which extent the inaccuracies may affect our results. We exclude the texts of which the language is unidentified. These texts include punctuations, emoticons and universal phrases, which accounts for a small proportion, but may still have an impact.

It is inherently difficult and ambiguous to rate a given sentence's sentiment, and even more so when the dimension of sentiment is only binary. Oftentimes, people use negative words to be humorous or sarcastic, which could be counted as "positive". Sentiments also heavily depend on their contexts. Future studies can utilize tools of context-based multi-dimensional sentiment analysis.

Finally, we did not impose upper bounds in the mining of vertical and horizontal comparisons of multi-attribute subgroups. For datasets with many attribute values, this could result in comparisons with many items that are difficult to interpret. Also, the amount of mined subgroup comparisons can be large, which becomes difficult to inspect. Besides taking measures to reduce redundancy algorithmically, another approach is to visualize these comparisons. Though a visualization tool tailored to this type of visual mining is yet to be developed, we can draw experience from the development of FreeBu and D-Explorer. And we consider this work to be the first step towards building a generic exploratory visualization tool that help users see the big picture of aggregated OSN data.[15]

7.7 Conclusion

We take an interdisciplinary approach towards mining OSN sentiment patterns. We investigated the sentiment differences across privacy levels and demographic factors. We find that not only the "conventional" or "stereotypical" hypotheses on demographic groups' sentiment expression are challenged, but more importantly, there are more detailed "stories" to explore and tell. For example, we find that the [17,20] group wrote less negative texts in *chats* than older age groups, which counters our hypothesis that late-teens have more negative texts. Furthermore, while most social data analysis focuses on publicly available texts, we see different sentiment expressions from users under different privacy settings. It reminds us that people naturally adjust their communication

[15] We are currently working on the extension of the paper [70], which is invited to be submitted to the journal of Social Network Analysis and Mining.

with others according to the size of the audience, among many other factors. Investigating these differences will improve our understanding of the data. For example, we find that the texts posted publicly are in general more positive than those posted privately, but the texts with a complete *public* setting are more reserved. Finally, using the subgroup-discovery paradigm, we present an approach with two algorithms that generalizes single-attribute testing, so as to provide more detailed insight into the relationships among different attributes, reveal interesting attribute-value combinations with distinct sentiments, and provide novel hypotheses for examination in future studies.

Chapter 8

Rediscovering Patterns in Discrimination-aware Mining

In this chapter, we focus on aggregate data transparency in the context of Discrimination-aware Data Mining (DaDM), on a bank-loan dataset. More specifically, historical data about people applying for loans from bank is aggregated, and potentially discriminatory decisions could be made against individuals in future bank-loan applications. We leverage existing DaDM techniques and develop a tool named D-Explorer, to help people, including both decision makers and loan applicants, explore and better understand discriminatory patterns.

8.1 Related Work

8.1.1 Discrimination-aware Data Mining

Pedreschi, Ruggieri and Turini [138, 142] introduced the problem of DaDM. Discrimination is the illegal use of the data on specific demographics as the basis for a decision. The instances of *directly discriminatory classification rules* and *indirectly discriminatory classification rules* are studied. For example, not giving a credit because of the applicant is a foreigner is directly discriminatory against foreigners. Not giving the credit because the applicant lives in a certain ZIP code, when it can be inferred that most people living there are migrants, is indirectly discriminatory against foreigners or migrants.

DaDM reveals such (direct or indirect) unfair treatment towards people in historical decision-making records by mining a set of classification rules. It also constructs a series of interestingness measures that characterize the degree of discrimination of the rules. These measures build on the *lift* measure of rules, which declare rules to be (potentially) discriminatory only if they exceed this threshold. DaDM computationally formalizes the legal notion of discrimination as a "disproportionate burden" put on certain demographic groups. For example, the U.S. Equal Pay Act states that "a selection rate for any race, sex, or ethnic group which is less than four-fifths of the rate for the group with the highest rate will generally be regarded as evidence of adverse impact". Discrimination-aware data mining has has since been extended in various directions, cf. [171, 33, 86].

8.1.2 DCUBE

The DCUBE system[1] implements the DaDM approach using the Apriori algorithm for extracting rules and an Oracle database for storing them [143]. Like DCUBE, we use the *German Credit* public domain dataset [5] for demonstration. Its attributes comprise various demographics (gender, marital status, nationality, ...) and details of the applicant's existing property and loan purposes. The Credit Class indicates whether a loan was given or not.

In DCUBE, the user can declare items as Potentially Discriminatory (PD) and set other parameters (max/min support, max size of frequent itemsets, ...) before the rule extraction process. For example, many laws forbid discrimination based on nationality, gender, marital status or age, which in the German Credit dataset are expressed by items such as

$foreign\_worker = yes$, $personal\_status = female\_div\_sep\_mar$ (a currently divorced, separated or married woman) and $age = gt\_52d6$ (a person older than 52.6 years). The remaining items are automatically taken as PND (Potentially Non-Discriminatory). The classification rules are then mined through both Direct and Indirect Discrimination analyses (DD resp. IDD), as shown in Figure 8.1.

DD rule mining yields rules in which an outcome (such as $credit = bad$) follows from potentially discriminatory items (usually in conjunction with PND items), while in IDD analysis, it follows from innocuous-looking PND items, which are shown to lead to inferences towards DD rules via background knowledge.

Via SQL queries, the user can extract a number of PD rules with defined constraints. A PD rule takes the form: $A, B \rightarrow C$, with A as PD item(s), B as PND item(s), and C being the class. Rules resulting from DD or IDD analysis

[1] http://kdd.di.unipi.it/dcube

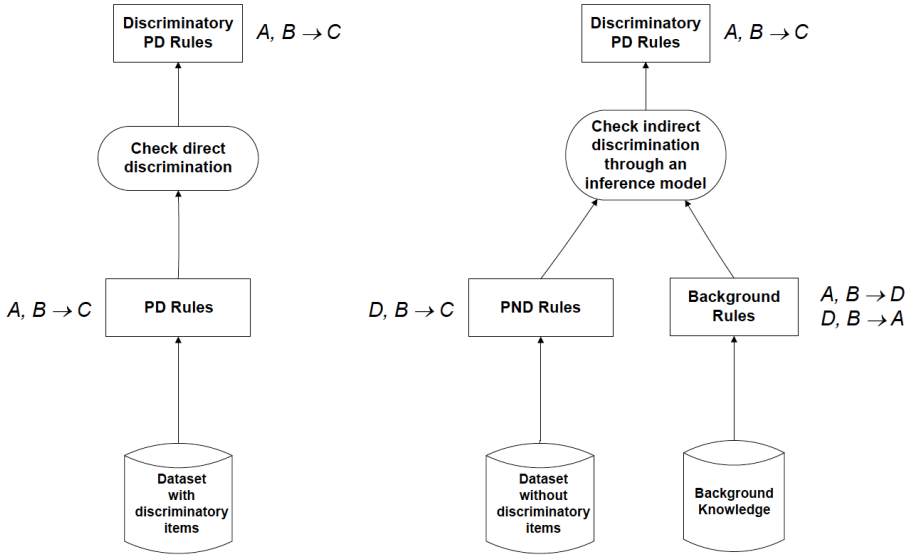

Figure 8.1: The modeling process of direct (left) and indirect (right) discrimination analysis [138].

Table 8.1: List of example PD rules $(A, B \rightarrow C)$

A (PD set)	**B (PND set)**	**C (class)**
$foreign\_worker = yes$	$own\_telephone = none$, $purpose = new\_car$	$credit = bad$
$personal\_status = female\_div\_sep\_mar$	$employment = from\_1\_lt\_4$, $property\_mag = real\_estate$	$credit = bad$
$personal\_status = female\_div\_sep\_mar$, $foreign\_worker = yes$	$age = le\_30d2$, $job = skilled$, $property\_mag = real\_estate$ $employment = from\_1\_lt\_4$	$credit = bad$

are ranked with respect to the interestingness measures that show the "degree of discrimination". Table 8.1 shows a list of example PD rules.

There are two limitations regarding the usability of DCUBE. First, the analysis via SQL queries is fully functional but appropriate for professional/technical users and much less accessible for other users. Second, the large number of rules makes it difficult to obtain an overview and interpret the patterns. For example, it would be difficult to find out how frequent items are within rules,

which two (or more) items are closely related to one another, which cluster of rules has large discriminatory measures, etc. To be able to understand these patterns better, derive new knowledge or new hypotheses, it is often helpful to have a meta-level view. Exploratory Visualization is a prime technique for achieving such condensed, meta-level representations of mining results.

8.1.3 Exploratory Visualization for Rules

Many visualization solutions have been proposed for association rules. For example: the 2D matrix with 3D cubes [185], in which the x and y-axes of the 2D matrix denote premises and conclusions of the rules respectively. The heights of the 3D cubes denote the interestingness measures of the rules, e.g. confidence. Alternatively, we can use each column of the 2D matrix as a rule and each row as an item. If a rule contains a certain item, this item's space in the matrix will be occupied with a 3D cube, whether this items belongs to the rule's premise or conclusion will be differentiated with color coding on the cube. Another example are the directed graphs of [24], in which rules are represented as graphs. Each node is an item or a combination of items [25], arrows are used to link from premises to conclusions. Other examples are the Mosaic Display [24] and ARVis [108, 25], etc. These visualizations can clearly express each rule's internal structure, but are insufficient to provide higher-level knowledge, especially when the number of rules gets large. The distribution of the rules and items or the inter-relations among them cannot be seen. Our approach of visualizing classification rules targets these shortcomings.

8.2 Meta-level Measures of Interestingness

The goal of our visualization is to highlight particularly "discriminatory" items on the one hand and to show relationships among the items and rules on the other. We therefore define new, higher-level measures of interestingness based on the discriminatory measures (*D-measure*) on individual rules in discrimination-aware data mining. A D-measure can be defined as Extended Lift [138], among other similar measures.

Let $conf(\boldsymbol{X} \rightarrow \boldsymbol{Y})$ be the confidence of the association rule $\boldsymbol{X} \rightarrow \boldsymbol{Y}$, and $\boldsymbol{A}, \boldsymbol{B} \rightarrow \boldsymbol{C}$ be an association rule such that $conf(\boldsymbol{B} \rightarrow \boldsymbol{C}) > 0$. The Extended Lift of the rule is with respect to $\boldsymbol{B}$ is defined as:

$$\frac{conf(\boldsymbol{A}, \boldsymbol{B} \rightarrow \boldsymbol{C})}{conf(\boldsymbol{B} \rightarrow \boldsymbol{C})} \tag{8.1}$$

$\boldsymbol{A}$ is the PD itemset, $\boldsymbol{B}$ is the PND itemset that is called *the context*, and $\boldsymbol{B} \rightarrow \boldsymbol{C}$ is call ed the *base-rule*.

The larger the *D-measure* of a rule is, the more certain we are of this rule being discriminatory against people in disadvantaged groups, such as foreign workers, single mothers, etc. Selection lift (*slift*) is chosen as the D-measure for assessing direct discrimination: the confidence of the rule with PD item(s) A, divided by the confidence of the rule whose premise contains $\neg A$. Extended-lift-lower-bound (*elb*) is chosen as D-measure for assessing indirect discrimination, which adapts *elift*, a variation of *slift*, with the confidence information from the background knowledge [142].

We combine this with established distribution-based measures of item(set) interestingness: Based on the *D-measure*, we define the item-focused *AD-measure* of the accumulated degree of discrimination of an individual item, and Mutual Information as a measure of correlation between pairs of items, based on the discriminatory rules these items are involved in. We also combine the rule-focused *D-measure* with a similarity measure on rule pairs to relate several rules to one another in their degree of discrimination.

Item Focusing: The goal of item focusing is to determine the degree of discrimination in individual items and to be able to relate these to one another in order to gain a meta-level view.

A simple measure is the *supp (support)* of an item or items in the mined rule set. Let an item be characterized as the equality of an attribute q to a value (range) v, let N be the number of rules, and $N(x)$ the number of rules satisfying the argument x. Let $Q = V$ denote an itemset: $q_1 = v_1, ..., q_m = v_m$ for $m \geq 1$. Then

$$supp(Q = V) = \frac{N(Q = V)}{N} \tag{8.2}$$

This measure does not take into account how discriminatory the PD rules are; therefore we define the *AD-measure* as a form of averaged *support*, which is weighted by the D-measure of these rules. Let R_i be the left-hand-side of rule i and $D\text{-}measure_i$ its D-measure, then:

$$AD - measure(Q = V) = \frac{\sum_{i=1}^{N} D\text{-}measure_i \cdot b(Q = V, R_i)}{supp(Q = V)} \tag{8.3}$$

$$with\ b(Q = V, R_i) = \begin{cases} 1 & if\ R_i\ contains\ Q = V \\ 0 & else \end{cases}$$

The AD-measure's range is $[0, \infty)$. In the visualizations, we only show the AD-measure for single items.

Other item-focusing interestingness measures have been proposed such as the magnitude-based and association-based interestingness functions in [22] and the summary-based interestingness functions in [94], which are attribute-oriented – the probability distribution of all the items within one attribute is taken into account. However, the attribute-oriented measures are limited in the sense that although the associations between two or more attributes (e.g. $foreign\_worker$ and $purpose$) can be well captured and ranked, the association between two or more items is ill-measured (e.g. $foreign\_worker = yes$ and $purpose = new\_car$). When an attribute only contains one value in all records, as in most of the rules extracted by DCUBE, the attribute-oriented measures would take it as a non-interesting item (see section 2.2.3 in [22]). In such a case, the items need to be treated individually, in other words, each item is taken as a non-dividable entity without the context of attributes, i.e. item-focusing. The support measure is in accord with this criterion, and so is *Mutual Information (MI)* [164] on pairs of items in the rules. We can use MI to characterize the interdependency between two items:

$$MI(q_1 = v_1, q_2 = v_2) = \log \frac{p(q_1 = v_1, q_2 = v_2)}{p(q_1 = v_1) \cdot p(q_2 = v_2)} \quad (8.4)$$
$$with\ p(Q = V) = N(Q = V)/M$$

and M being the total number of items in all rules (including overlaps).

Interestingness measures can be categorized into objective and subjective measures. Objective measures such as *supp* and *MI* focus on the probability distribution of data. Subjective measures take a user's judgement and experience as a part of the measurements, such as exceptions [75] and expectations [116], which involve a user's predefinition of a specific set of items. The "Discrimination Discovery in databases" approach also lets a user subjectively select PD items. Thus, *AD-measure* is a hybrid objective/subjective measure.[2]

Rule Focusing: The support-confidence framework can be insufficient to evaluate the quality of a rule [14]. Many rule-focusing measures of interestingness have been proposed, including objective measures such as rule-interest, lift, conviction, Loevinger index, implication intensity, coverage, strength, performance, Sebag and Schoenauer index and IPEE, and subjective measures such as simplicity, unexpectedness and actionability[116, 24]. The D-measures in the DCUBE system also contain subjectiveness in that they take the user's

[2]Like DCUBE, we disregard accuracy-based measures of rule interestingness. Integrating such aspects is an important direction for future work.

prior knowledge of the data domain into account. We will use the D-measures (*slift* and *elb*) to characterize each rule's interestingness.

To characterize the similarity/distance between two rules, we concentrate on all the items in the rule (for association rules) or all items on the left-hand-side of the rule (for classification rules) and regard them as an itemset or vector in the space spanned by all attributes. We can then apply a number of measures such as the Hamming distance, Jaccard distance, Dice's coefficient, or Cosine similarity and its binary version, the Tanimoto coefficient. Another measure of the association between two (or more) rules is through fuzzy meta-association rule extraction [187].

We will use the Jaccard distance to indicate the degree of dissimilarity between two rules. In contrast to other measures, it is applicable in a straightforward way to the sets with different items as well as readily implemented. Let $S(R)$ denote the itemset of R. Then the Jaccard distance between R_1 and R_2 is

$$J(R_1, R_2) = 1 - \frac{S(R_1) \cap S(R_2)}{S(R_1) \cup S(R_2)}. \tag{8.5}$$

8.3 D-Explorer

D-Explorer, our tool for visualizing the degree of discrimination inherent in rules and items, is built in Java SE6, mainly with Processing-1.2.1 for graphical design[3]. It also uses packages from Weka-3.6.4[4] and Jtreemap-1.1.0[5].

The PD rules in DCUBE can be directly extracted from the database via a JDBC connection in the tool, or we can extract the rules in advance, which are then stored in on the local machine for the tool to read[6]. The latter is recommended because querying the database can be time-consuming, which inhibits repetitive querying. In order to accurately reflect the general information "trends" or distribution of the rules from a high level, we need not only a large quantity of rules, but also good quality (highly discriminatory). Based on these two criteria, two sets of PD rules are extracted from DCUBE for visualization. On the German Credit Dataset, this led to one set of 1062 rules extracted in DD analysis, with minimum $support > 20$ and $slift > 2.6$, we call it Rule Set 1 (RS1). The other set contains 770 rules extracted in IDD analysis, with minimum $support > 20$ and $elb > 1.3$, we call it Rule Set 2 (RS2).

[3] http://processing.org/
[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://jtreemap.sourceforge.net/
[6] In our demonstration, two .csv files of rules are used

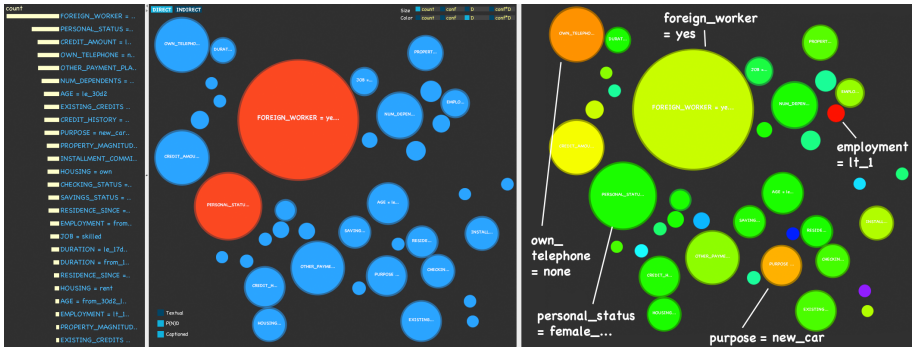

Figure 8.2: Visualization with bubbles on RS1.

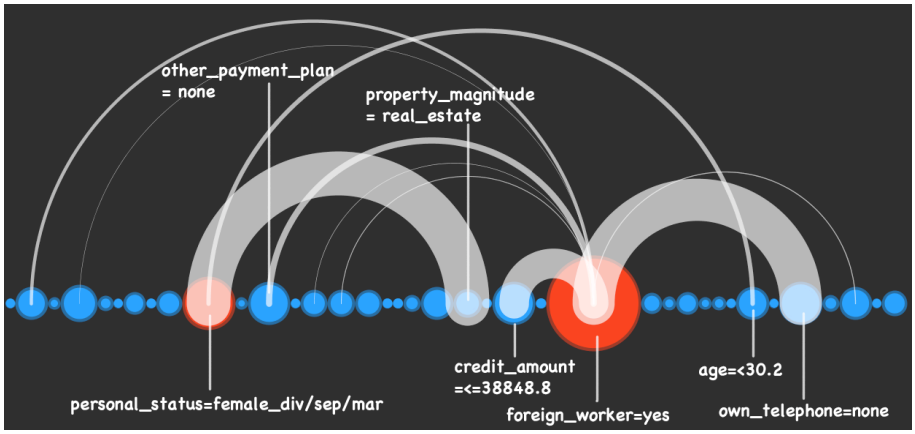

Figure 8.3: Associations between items in the Bubble View on RS1

Item Oriented: Figure 8.2 shows the interface of the tool. On the left is a panel for item information. The lengths of the bars are scaled and sorted according to the supports of the items. In the middle is the view of bubbles, each bubble represents an item, of which the size is scaled according to its support. In the middle, colour indicates PD (red) and PND (blue); on the right of the figure, rainbow colors indicate the AD-measure. The rainbow-colors scale ranges from red (high) via orange, yellow, green, and blue to purple (low). From Figure. 2, we get an overview of the distribution of the items in RS1. The items $foreign\_worker = yes$ and $personal\_status = female\_div\_sep\_mar$ are very frequent, because almost every rule contains at least one the two PD items. We also see that the PD item $age = gt\_52d6$ is not frequent, which implies that the assumption that old people are often discriminated

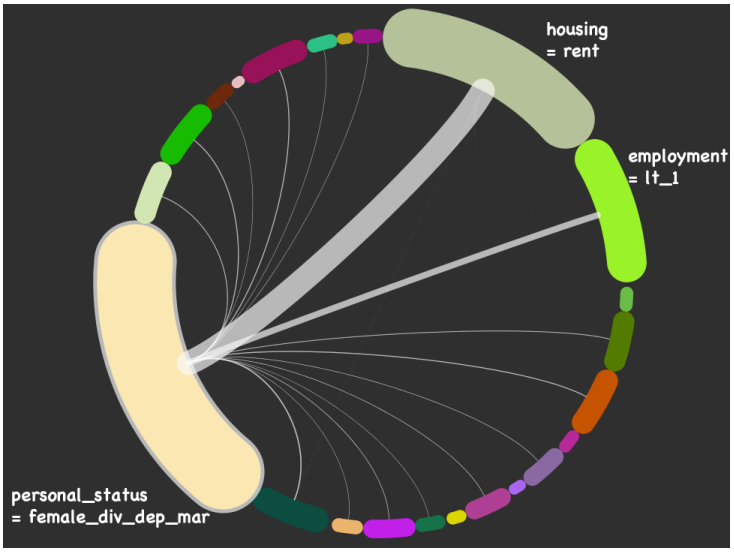

Figure 8.4: Associations between items in the Arc View on RS2

in various situations is not true. However, besides the two PD items, we also see other major items in the RS1, such as *other_payment_plans* = *none*, *credit_amount* = *le_38848d8*, *own_telephone* = *none*, etc. On the right, we see that although items such as *foreign_worker* = *yes* and *personal_status* = *female_or_div_or_sep_or_mar* are larger in size, their AD scores are not high (yellow or green) compared to items *employment* = *lt_1*, *own_telephone* = *none* and *purpose* = *new_car* (red or orange). In terms of AD-measure, the red or orange items appear more "effectively" in the rules than the yellow or green ones. Especially for *employment* = *lt_1*, the item appears not frequently in general, but quite frequently in the rules with high discriminatory scores, which shows that a person in a disadvantaged group who has less than one year working experience is often rejected by the bank on a loan, hence, discriminated.

The relationships between items are explored next. In Figure 8.3, the bubbles are aligned and connected with semi-circles, of which the weights are determined according to the pairwise Mutual Information (MI) between items. The threshold of MI in Figure. 3 (a) is 0.06 (half the maximum MI value). We see among all the associations above the given threshold, the item *foreign_worker* = *yes* is strongly related to *own_telephone* = *none* and *credit_amount* = *le_38848d8*, which means the two combinations appear more often and are more influential than the others. We can interpret this

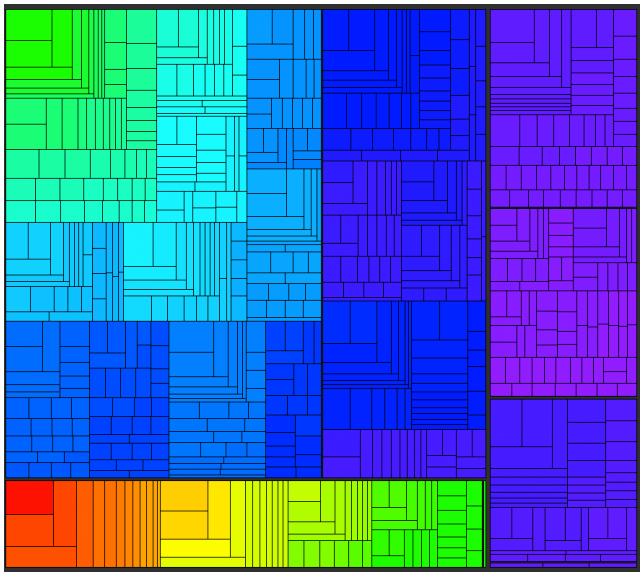

Figure 8.5: The treemap view on RS2

as follows: a foreign worker is often discriminated when (s)he does not own a telephone and/or has less than 38848.8 units of credit. We also discover that $personal\_status = female\_div\_sep\_mar$ has a strong connection with $property\_magnitude = real\_estate$ and $age = le\_30d2$. This indicates that non-single women tend to be discriminated against when under 30 years old or when owning real estate, which is a surprising discovery.

An alternative visual presentation of the association between items is shown in Figure 8.4. This time we investigate the ruleset in RS2. Each arc in the circle represents an item. The weights of the arcs are scaled according to the counts of items. The white curves connecting different arcs represent associations between items, and their weights are also based on MI, as the semi-circles in Figure 4. Thicker connection means stronger association. When the user hovers the mouse over a certain item, the relevant connections will be shown, the others are hidden. As we observe through visualizations of the rules extracted from indirectly discriminatory analysis (which performs background checking), new knowledge is discovered: we find new strong links between $personal\_status = female\_div\_sep\_mar$ and two PND items: $housing = rent$ and $employment = lt\_1$. This indicates that a non-single woman tends to be discriminated against if currently renting a house or having an employment experience of less than one year.

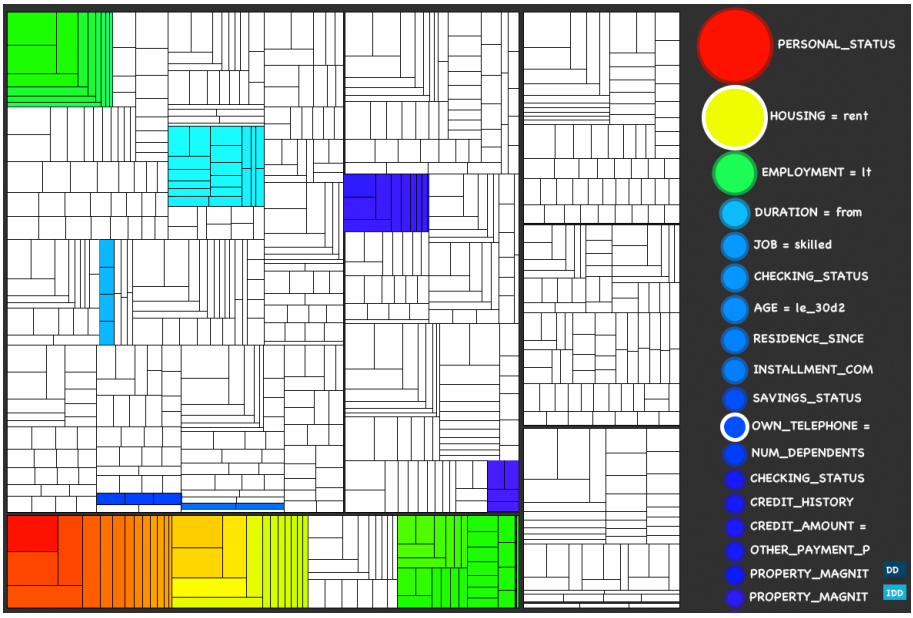

Figure 8.6: The treemap view on RS2 with filtering

Rule Oriented: Rule-Oriented visualization shows an overview as well as a general distribution of all the PD rules, how some (combinations of) items affect this distribution, and whether there is an underlying pattern with respect to D-measures.

The hierarchical clusters of rules are built based on the pairwise Jaccard distance of the rules, with each rule as a leaf in a dendrogram. We use agglomerative hierarchical clustering [59], with the linkage criterion Weighted Pair Group Method of Arithmetic Average. This puts the rules with similar items into the same or adjacent branches in the hierarchy.

Then, we use the Squarified Treemap [99, 32] space filling approach to visualize the dendrogram of rules, with each elementary rectangle as a PD rule. An elementary rectangle is weighted as well as colored with rainbow colors based on the D-measures, so that a rule with higher discriminatory score is more reddish, the one with lower score is more purplish. Figure 8.5 shows a treemap visualization on the PD rules extracted in the IDD analysis. We see that on the left, the rules are clearly clustered into distinct regions and the D-measure distribution is strongly correlated with the distribution of the regions (since we color the rectangles based on the rules' *elb* scores), which implies that certain combinations of items within the rules lead to more discriminatory situations

than others. E.g. the region at the bottom starting from the left (red) and the region at the top-left corner (green) contain rules with a relatively high D-measure, which should inspire further investigation.

The items are aligned on the right hand side of Figure 8.6, sized according to their supports. User can select multiple items at a time; the rules (rectangles) not containing the selected items will be colored white. We can see that the remaining colored areas contain the items $housing = rent$ and $own\_telephone = none$, which means this combination of these items quite often appear in high-ranked PD rules.

8.4 Conclusion

In this chapter, we detailed the developement of an exploratory visualization tool named D-Explorer. We examined a specific use case in the field of DaDM. An exploratory visualization approach complemented the original textual, classification-rule enumeration approach. D-Explorer can be generalized to visualize other rules.

Berendt and Sören [20] conducted a user study, showing that D-Explorer's form of displaying the results of DaDM for further exploration supported correct interpretations of the data and useful behaviour in a realistic scenario. The participants in this user study were asked to interpret static visualizations of D-Explorer. It is worth investigating how users interact with the tool. We further identify the design flaws and explore other visualization options for the tool in Chapter 10.

Part III

Practical Visualization Design

Chapter 9

Investigating Online Data-visualization Libraries

Previously, we have developed several exploratory visualization tools. These developments would not be possible without the advancement of browser technologies and data-visualization libraries. The libraries assist designers and developers in both academia and industry to build complex visualization systems. Good libraries can significantly reduce the unnecessary time and energy consumed in repetitive coding and subsequent maintenance, and liberate designers/developers to focus on visualization itself. But in academia, most effort is concentrated on novel visualization design or related user studies. The implementation side of online visualizations receives little attention. In this chapter, we study current Javascript data visualization libraries and provide a systematic approach to compare and evaluate them. More specifically, we examine the libraries in terms of application domains, abstraction levels, visualization tasks and design patterns. We translate visualization task taxonomies to library implementation requirements in order to comprehensively capture different aspects of data-visualization development. Furthermore, we use code snippets to demonstrate different design paradigms and explain the corresponding strengths and weaknesses. We propose various improvements for data-visualization development.[1]

[1]This chapter contains a paper [68] that is currently under review for IEEE Transactions on Visualization and Computer Graphics.

9.1 Motivation

With the advancement of web browsers, researchers and practitioners can build complex, interactive and large-scale online visualizations. However, in both academia and industry, most effort is focused on designing new visualizations, developing visualization tools and studying how users interact with visualizations, whereas the implementation aspect of data visualization is largely ignored. To create any modern software, including data-visualization systems, one must rely on libraries, which, to various extents, hide low-level implementation details and simplify the coding process.

Recently, we see a boom of new online data-visualization libraries, but the landscape of such libraries remains obscure to data-visualization developers. It becomes difficult to choose a library that is just "right". This is due to the large quantity of available libraries, and library viscosity[2] [77] and incompatibility. Furthermore, data-visualization development is a dynamic process, it gathers prior experience and produces more powerful libraries, ongoingly. At this point, we need to summarize existing libraries, and create a connection between the libraries and their technological ecosystems, visualization tasks and design patterns. Only in this recap could we identify new directions and move forward. In this chapter, we establish a comprehensive approach to compare and evaluate Javascript data-visualization libraries, so that, (1) a visualization developer can appropriately choose an existing library that fits his needs, (2) a library developer knows how to improve or extend an existing library, (3) a library developer has a set of guidelines to develop new data-visualization libraries.

Client-side & Server-side: The web is based on the client-server architecture. The user uses a client-computer to send requests to a server-computer, in response, the server sends back the requested content and program to the client. Server-side visualization libraries concentrate computations on the server, send back only the "presentation" for the browser to render. This requires an extra round of translation from server-side code to client-side code. It also produces more communication overhead. Another disadvantage of server-side libraries is the computational bottleneck of a server handling many concurrent users.[3] But thanks to the advancement of both hardware and software in personal computers, clients can now handle more demanding tasks than before. This makes client-side technologies flourish, and leads us to further investigate client-side libraries for data visualization.

Rise of HTML5 and Javascript: There are mainly four categories of

[2] People resist learning unfamiliar, new libraries.

[3] For example, as the report shows in `http://java.dzone.com/articles/performance-report-server-side`.

technology for client-side Rich Internet Applications (RIAs): Adobe Flash, Microsoft Silverlight, Java-based and Javascript-based. Adobe Flash[4] is a multimedia platform that can run RIAs. Applications can be programmed in ActionScript. Browsers need to have Adobe Flash Player pre-installed as a plug-in. Microsoft Silverlight[5] is an RIA library, a plug-in is also required. Java-based libraries such as Java Applet and JavaFX[6] rely on Java Virtual Machine (JVM) to run. Applications are programmed in Java. Javascript-based libraries do not require another run-time environment, all modern browsers support Javascript by default. The upgrade of Javascript engines in major browsers and the wide adoption of W3C standard HTML5 further increase the popularity of Javascript. As of July, 2015, 89.8% of websites use Javascript, 10.6% Flash, 0.1% Silverlight and 0.1% Java, and the usage of Flash, Silverlight and Java on websites has been declining, while that of Javascript remains dominant[7]. From Google Trends, we can also see that HTML5-based technologies have been in the process of replacing others in the past decade[8].

9.2 Related Work

We have seen studies on the evaluation and comparison of non-Javascript libraries [111, 109], or existing visualization tools [107]. While tools can be directly used by end-users to create visualizations, they do not allow development of new visualization systems or designs. [88] investigated a wide range of open-source visual analytics libraries in terms of their visualization and analytics features, which includes three Javascript-based libraries (see Table 9.10 and 9.11): Google Charts, Javascript Infovis Toolkit and Protovis. Protovis is no longer under active development. Instead, its improved successor d3 [27] has gained popularity. Besides the aforementioned Javascript-based libraries, many other ones exist. They should be studied, compared, evaluated and summarized. More importantly, there has been no study that connects data-visualization libraries with their technological ecosystems, visualization tasks and design patterns. We bridge this gap.

[4] https://en.wikipedia.org/wiki/Adobe_Flash

[5] https://en.wikipedia.org/wiki/Microsoft_Silverlight

[6] https://en.wikipedia.org/wiki/JavaFX

[7] http://w3techs.com/technologies/overview/client_side_language/all.

[8] http://www.google.com/trends/explore#q=adobe++flex,microsoft+silverlight,java++applet,HTML5,gwt.

9.3 Library Overview

In this section, we give an overview of current Javascript-based visualization libraries. We categorize these libraries based on their application domains, rendering technologies and abstraction levels.

9.3.1 Application Domains

In total, we identified 104 libraries (Table 9.10 and 9.11.) based on Google search with the term "javascript visualization library".[9] 86 libraries were released or updated after 2013. The majority of the libraries specialize in one or two domains. Table 9.1 lists the domains and the corresponding library counts. We see that chart-visualization is a common theme, followed by maps and graphs. They correspond to the three most common data types that we encounter daily – tabular, networked and geospatial data. Chart visualizations include line, bar, area, radial and bubble charts, etc. Map visualizations are thematic maps[10] showing specific geographic areas combined with texts, colors and geometric shapes. Most libraries focus on 2D drawing, because 2D visualizations are usually sufficient for conveying information. Also, it is well known that the depth cues in our visual system include occlusion, perspective distortion, etc. [130]. But 3D data visualizations can be useful in computational plotting[11], more specifically in mesh rendering and physics simulation, which are common in scientific visualization. Color-specific libraries provide color templates, conversion and scaling utilities in different color spaces.

Moreover, we consider "data visualization" an umbrella term that covers both "information visualization" (*infovis*) and "scientific visualization" (*scivis*). Though there is no clear boundary between *infovis* and *scivis*, we can characterize the former as visual representations for abstract data and the latter for concrete, physical data, often in a three-dimensional format (e.g. a medical scan of human body). Also, as map visualizations usually emphasize the abstract information on top of the geographic layer, we consider them *infovis*. Most libraries are developed to accommodate *infovis*. This phenomenon comes naturally as the field inspires creation of new forms of visual design and user interaction. Its central concern is to determine whether the chosen or newly designed visualization is suitable for the data and tasks at hand [130]. We therefore focus on the *infovis* libraries in the rest of the chapter.

[9] We exclude the native APIs such as Canvas, WebGL and SVG from consideration, as the purpose of libraries is to overcome the cumbersomeness in using native APIs directly.

[10] `http://en.wikipedia.org/wiki/Thematic_map`.

[11] For example, Toxiclibs.js uses the 3D library three.js to fulfill some of its functionalities.

Table 9.1: Library Domain Categories

Domain	Description	Freq.
chart (*infovis*)	line/bar/pie/area/radial charts, etc.	65
map (*infovis*)	choropleth/symbol maps, etc.	21
graph (*infovis*)	networks or tree structures	18
general (*infovis*)	general-purpose	8
text (*infovis*)	word cloud	5
3D (*infovis*)	3D drawing	3
color (*infovis*)	color preset and conversion	3
comput.plot. (*scivis*)	mesh/wave generation, physics, etc.	2

9.3.2 Rendering Technologies

There are two major rendering technologies used by the libraries – HTML5 Canvas (Canvas for short) and SVG[12]. The former was developed by the Web Hypertext Application Technology Working Group (WHATWG), and standardized by World Wide Web Consortium (W3C), the latter was developed by W3C. Both are 2D drawing methods that are manipulatable via Javascript. The difference is that: Canvas renders raster graphics, where visualization is realized within the HTML5 canvas element in the Document Object Model (DOM)[13]; SVG is an XML-based vector graphics format, of which each visual object itself is an indexed DOM element. Thus SVG visualizations can be directly programmed in HTML tags and styled by Cascading Style Sheets (CSS).[14] Canvas directly draws an un-retained bitmap to the screen. SVG keeps the visual objects in the DOM, then draws the corresponding bitmap. Both technologies have corresponding low level libraries and Application Programming Interface (APIs). Among the 104 libraries, 32 support only Canvas, 48 support only SVG, and 19 support both. Other libraries use WebGL to render 3D graphics, or HTML tags to render simple diagrams.

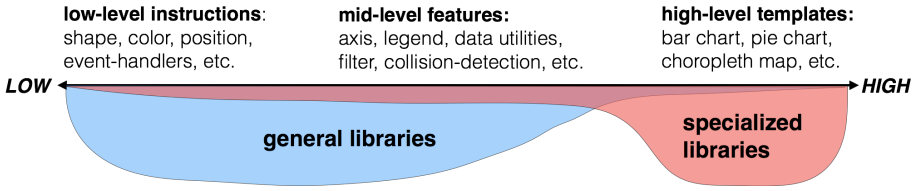

Figure 9.1: Different abstraction levels of data-visualization libraries

9.3.3 Abstraction Levels

There is a range of abstraction levels on which libraries can operate. As shown in Figure 9.1, low-level instructions/utilities draw basic visualization elements such as shapes, colors, and handle basic events such mouse, key events. Mid-level features offer more advanced visual components such as axes, filters and collision-detection. Finally, high-level templates directly draw popular visualizations such as bar chart and choropleth map. Following Munzner [130], we call these visualizations "(visualization) idioms". Developers do not have to define each constituent of an idiom. The templates manage the drawing details and expose configuration options for customization.

As shown in Figure 9.1, "specialized libraries" mainly support building visualization idioms, but can also provide low-level or mid-level features. The majority of the libraries are specialized libraries. They focus on the development of chart, map and graph idioms, as shown in Table 9.1. "General libraries" provide comprehensive low-level instructions/utilities, and/or mid-level features. Sometimes high-level templates are also offered. Developers can use these libraries to build novel or extensively customized visualizations. From Table 9.1 we see that only eight libraries are genera-purposel libraries.

In Section 9.4, we will examine the general libraries on the low abstraction level. We focus on the specialized libraries for visualizing idioms in Section 9.5. In Section 9.6, we study the mid-level features required for bridging the gap between the two ends of the abstraction spectrum. We will only examine the libraries with a free licence, which allows the open access to the APIs and/or source code.

[12]Browsers nowadays all natively support SVG and Canvas, see `http://caniuse.com/#feat=svg` and `http://caniuse.com/#feat=canvas`.

[13]`http://www.w3.org/DOM/`

[14]`http://www.w3.org/Style/CSS/Overview.en.html`

Table 9.2: Architectural Choices of General Libraries

library	renderer	syntax	scope	scene graph	native access
easel	Canvas	JS	DOM	self	✓
fabric	Canvas	JS	DOM	self	✓
kinetic	Canvas	JS	DOM	self	✓
paper	Canvas	JS	DOM	self	✓
processingJS	Canvas	Java/JS	DOM	N/A	✓
bonsai	Canvas	JS	self	self	×
d3	SVG	JS/CSS	DOM	DOM	✓
raphael	SVG	JS/CSS	DOM	DOM	×

9.4 General Libraries in Low Abstraction

9.4.1 Architectural Choices

We first list the architectural choices of the general libraries in Table 9.2. These choices influence the corresponding API designs. The libraries easel, fabric, kinetic, paper, processingJS and bonsai are all Canvas-based. processingJS has a distinct syntax. Unlike the others with the JS syntax, its main syntax is Java-like. processingJS is the Javascript port of Processing.[15] Processing is a programming language and library for data visualization and electronic arts, running on Java Virtual Machine. The syntax of Processing can be considered as simplified Java. The goal of processingJS is to render original Processing code in web browsers with Javascript engines. Programmers can still write Object-Oriented (OO) Processing code, processingJS subsequently translates the code into native Canvas instructions in Javascript. d3 and raphael use SVG as renderer. CSS could also be used to modify the appearance of SVG elements.

The scope of a library indicates "how far the instructions of the library can reach" within the browser. Ideally it should be DOM, within which the browser itself operates. Because oftentimes, a visualization needs to interact with other DOM elements, such as an input field, a checkbox, to fulfil its functions. We see all but bonsai have the DOM scope. bonsai is constrained within its own

[15] `http://processing.org`.

Table 9.3: Low-Level Utilities for Basic Mark-Drawing and Event-Handling

marks	*shape*	symbols, path, text
	position	exact positioning, transform: translate
	size	exact sizing, transform: scale
	tilt/angle	transform: rotate, transform: skew
	color	color space, color format, color preset, filters, pixel array
	transition	transitions for shape, position, size, tilt/angle and color
events	*mouse*	click, move, scroll, enter, hover, leave, down, up, drag, drag-begin, drag-end,
	key	code, down, up
	touch	start, end, move, cancel

environment.[16]. This causes trouble to communicate with the "outside world", and severely limits the library. For this reason, we will exclude bonsai from further consideration and focus on the general libraries with the DOM scope.

Scene Graph is the data structure that a library uses to maintain and manage its data and visual objects. These objects can be updated, removed and sometimes individually event-handled. DOM is the scene graph of browsers. SVG-based libraries have the natural advantage of readily using DOM as their scene graphs. Canvas-based libraries, however, do not have such an advantage, as Canvas is just a single element in the DOM. The Canvas-based libraries easel, fabric, kinetic and paper have their own implementations of scene graphs, which offer retained trees of data and visual objects. Lastly, we look at whether a general library has the full access to the underneath, native Canvas or SVG APIs. The native APIs represent the complete range of drawing abilities, which can be directly invoked at convenience without encapsulation. We see that all but bonsai and raphael allow full native access.

[16]More specifically, the variables and functions of bonsai are only recognized within the "code" segment of the "bonsai.run()" function. Furthermore, within this self-scope, the native Javascript functions such as "console.log()" are not accessible. See `http://docs.bonsaijs.org/`.

Table 9.4: Comparison of General Libraries at Low-Level

	shape			*color*				
library	**symbols** (oval, circle,rect+)	**path**	**text** (basic+)	**space** (RGB,HSL/V+)	**color set**	**filters**	**pixel**	**transition & event** (basic+)
easel	star	add point			×	5	✓	t: object, e: object
fabric	triangle	add point			×	13	✓	t: object, e: object
kinetic	isogon	add point			×	17	✓	t: canvas, e: object
paper	triangle,isogon	add point			×	0	✓	t: canvas, e: object
processing	triangle	add point			×	8	✓	t: canvas, e: canvas
d3	triangle,square, diamond,cross	point array, interpolators	decorate, style, flex. position	LAB, HCL brighter,darker	categories, interpolators	21	×	t: object, e: object
raphael		add point	decorate, style, flex. position		×	0	×	t: object, e: object

9.4.2 Low-level Instructions/Utilities

The most basic elements of information visualization are marks [130], they are primitive geometric shapes, including points, lines/curves and polygons. Properties of marks can be chosen and varied based on the underlying data to achieve visual encoding. Following Munzner [130], we categorize the mark properties into six categories: shape, position, size, tilt/angle, color and transition. Note that we generalize the original category "motion" to "transition" to accommodate the fact that not only positional change through time, but also other changes, such as color or size change, can be taken as means of visual encoding. A general library needs to provide an API that modifies properties of marks. Moreover, a static visualization is of limited value, users need to interact with the visualization so as to browse, search and explore data. Thus, a general library also needs to enable developers to handle interaction events. The basic instructions and utilities to draw marks and handle events are summarized in Table 9.3. According to this table, we compare the seven general libraries — easel, fabric, kinetic, paper, processingJS, d3 and raphael, as detailed in Table 9.4. Note that because all the seven libraries provide the access to low-level instructions for exact positioning, exact sizing, and transforms: translate, scale, rotate and skew, these aspects are not shown in Table 9.4. Only the aspects that differentiate the libraries are shown, namely shape, color, transition and event handling.

Shape It is important for a library to provide a set of symbols that can be readily drawn to encode categorical data. All the libraries provide the basic shapes: oval, circle and rectangle, d3 offers a richer set of shapes than the rest. The "path" utility should also be provided for drawing arbitrary lines, curves and shapes. Most libraries follow a simplified SVG path[17] style, yet all the points

[17] `http://www.w3.org/TR/SVG/paths.html`

still need to be added individually to construct a path. d3 lifts this triviality by providing a set of interpolators and a more concise constructor `attr("d", points)` that takes a point array. The seven libraries also have all the basic text drawing utilities (font, text metrics). But SVG-based libraries have the advantage of scalable typography, text styling, decoration (e.g. underline) and flexible positioning (i.e. aligning texts along a path).

Color All the libraries support RGB (Red Green Blue) and HSL/V (Hue Saturation Lightness/Value) color spaces to define colors. Colors can be formatted in string names, hexadecimal or decimal triplets. While RGB is suitable for machine input, its color components make it difficult for humans to interpret. HSL/V spaces are more intuitive for describing how humans perceive colors. d3 further supports LAB and HCL color spaces[18], which are more human friendly than HSL/V in the sense that they are perceptually uniform[19]. Convenient functions `brighter()` and `darker()` are also provided for all the color spaces in d3. Moreover, color coding categorical or numerical data is a non-trivial task [180]. The number of distinguishable colors in a given set is rather limited, and need to be carefully chosen. Libraries should provide a pre-selected colors for categorical encoding. We see only d3 making this effort by providing a set of 10 colors and three sets of 20 colors. It also enables color mapping for numerical values because of its underlying interpolators, of which we will see more instances in the mid-level features in Section 9.6. Furthermore, we count the available pixel filters in a library, d3 with its native access, comes on top. kinetic and fabric also provide abundant filters. Finally, as the advantage of Canvas-based libraries, pixel manipulation is readily available via pixel array.

Transition & Event All the libraries support transitions and mouse/key/touch event-handling. The main difference lies in the "granularity" of the transitions and event-handling. Because Canvas occupies a single element in DOM, and it draws a bitmap that is immediately forgotten, the animation and event-listening apply to the entire canvas. Moreover, Canvas uses the `requestAnimationFrame()` function to realize animations. This function is repeatedly called to render the instructions inside. The Canvas-based libraries provide corresponding wrapper functions. However, the callback of `requestAnimationFrame()` by default is endless, a visualization developer has to manage both the scope and the duration of a transition. The SVG-based libraries d3 and raphael have the natural advantage that they operate as a part of DOM, which allows transition and event-listening on individual tags (i.e. objects). The Canvas-based easel and fabric provide object-oriented transition

[18]International Commission on Illumination (CIE)'s LAB and LUV color spaces. See the example in `http://bl.ocks.org/mbostock/3014589`.

[19]A system is perceptually uniform if a small perturbation to a component value is approximately equally perceptible across the range of that value.

Table 9.5: Visualization Idioms in Different Domains

chart idioms	scatterplot, dot&line chart, (stacked)bar chart, bubble chart, pie chart, ring chart, radar chart, streamgraph, heatmap, parallel coordinates
graph idioms	rectilinear node-link diagram, icicle, radial node-link diagram, concentric circles, nested/packed circles, force-directed graph, adjacency matrix, treemap, arc diagram
map idioms	choropleth map, symbol map, flow map
text idioms	word cloud

and event-handling as well. Individual transitions still have to be managed by the developer in kinetic and paper. processingJS adopts a fully imperative approach without individual transition and event-handling.

From Table 9.4, we see that d3 has a richer set of useful low-level utilities than the others. For example, d3 provides more symbols, more color spaces and the color sets for categorical encoding. It also leverages the underneath SVG APIs for better visualization expressiveness and control. We also learn that though most libraries have the basic instructions and utilities to modify the six aspects of marks and handle the three event types, utilities such as a richer shape preset, interpolators, categorical color pre-selections are still commonly missing. Moreover, Canvas-based libraries have the disadvantage of reimplementing a data structure (such as a scene graph) to accommodate object-oriented transition and event-handling.

9.5 Specialized Libraries for Idioms

Low-level utilities of visualization libraries, regardless how expressive and versatile, do not address the demand for fast delivery of reusable designs. Developers need to be lifted to a much higher level of abstraction to program visualizations, yet with satisfactory customizability.

Table 9.6: Taxonomy of Visualization Idiom Components

	encodings	**actions & interactions**
vis	chart/map/graph/word cloud	zoom, pan
parts	[*data*] incl. T/CSV, JSON, texts svg, geoJSON, topoJSON, tiles	import/parse, format
	[*marks*] with idiom encodings	arrange, highlight, transition
	[*axes*] incl. ticks, titles, grid	arrange, transition, axis-pointer, brush
	[*legends*]	arrange, filter

9.5.1 Towards Idioms and Idiom Components

Many forms of visualization, including visual encodings and interactions, have matured over the years, and are being widely used. They become idioms. Due to the large range of variations of an information visualization, and the difficulty to fully distinguish an idiom and a full-fledged visualization system, there lacks a formal definition of visualization idiom. We approach this empirically by listing major visualization idioms in Table 9.5.

We can see that the majority of the libraries try to directly support programming these idioms. It is worth studying how the high-level libraries work, what is common, what are the (dis)advantages and what remains to be improved. We summarize a catalog of visualization idioms based on [130] and [91]. More specifically, we consider the idioms unique in their visual encodings, and merge the "duplicates". For instance, the Index Charts, Small Multiples and Horizon Graphs in [91] are essentially line charts with or without "color fill", we therefore only consider the generic form "dot & line chart". We further add word cloud[20] to the catalog. The idioms are organized by the domains to which they typically apply, as summarized in Table 9.5.

We emphasize that modern visualizations should enable user interactions to accommodate (large) dataset exploration. Hence the visual encodings are often coupled with interactions, and sometimes interactions are part of the encodings

[20]Word cloud, or tag cloud, is used to show the key words of a given document in a 2D packed layout.

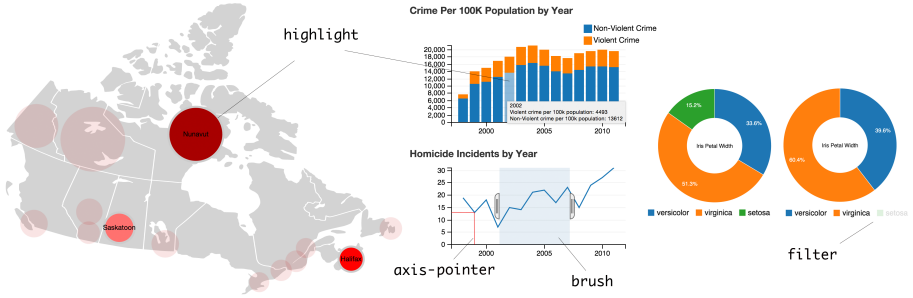

Figure 9.2: Example Interactions in Visualization Idioms

(transition). To support creating and customizing an idiom, a library needs to manage its encoding constituents and user interactions under the hood. The constituents are: data, marks, axes (if it is a chart) and legends (Table 9.6). The data facility of a library reads, parses, and formats the data if necessary, to prepare it for subsequent visualization. Common data formats should be supported, which include Tab/Comma-Separated Values (T/CSV) and JSON for charts and graphs, svg, getJSON, topoJSON and tiles for maps, and texts for word cloud. Marks, legends and/or axes should then be automatically mapped, as defaults, to the data with suitable properties and interactions. Examples of the interactions are shown in Figure 9.2.[21] Interactions on visual objects ought to be implemented by default, without the developer explicitly defining them. For instance, when the user mouse-hovers a visual object, such as a bar, that object is automatically highlighted with color contrast and/or balloon text. When he brushes an area in a chart, the corresponding sub-region is selected and displayed. An axis-pointer can be switched on to show the coordinate positions in the axes. The user can also click on the legends to filter a chart. Whenever a selection or filter is triggered, animations (i.e. transitions) are performed to show the gradual changes. On the visualization level, zoom and pan can be performed so that all parts of the visualization change synchronously and collaboratively. The (inter)actions that correspond to the encoding parts of idiom visualizations are summarized in Table 9.6. This table is informed by and closely related to the taxonomies proposed in [92, 31].

For all the libraries specialized in chart, map, graph or text visualizations, we summarize their supported idioms and key drawbacks in the columns “idioms” and “lack” respectively in both Table 9.10 and 9.11. The * sign indicates that a library supports all the idioms that are listed in Table 9.5. If the support for only a few idioms is lacking, e.g. the bubble chart idiom, we denote it

[21] screenshots from `https://dc-js.github.io/dc.js/crime/index.html` and `http://c3js.org/examples.html`

as ** − bubble* in the "idioms" column. Also, on the one hand, some libraries offer encapsulated idiom templates with customization options. This approach conveniently hides implementation details such as data parsing and mapping, interaction handling. However, sometimes, a template's coverage of encoding parts and (inter)actions is incomplete. For example, highlight or brushing is not supported (*no highlight* or *no brush*). We check, in accordance with Table 9.6, whether the encoding parts and corresponding (inter)actions are provided by the libraries. Missing parts and (inter)actions are indicated in the "lack" column. On the other hand, some libraries do not offer templates, but rely on low-level or mid-level instructions/features, with which the developer composes a visualization idiom. This approach can create a steep learning curve for developers, and make the code difficult to be reused. We denote these cases with *low-mid* in the "lack" column.

In the domain of charts, not surprisingly, dot & line chart, bar chart and pie chart are the most commonly supported idioms. Heatmap and parallel coordinates are the least supported idioms. Meanwhile, streamgraph, radar chart and ring chart are rarely supported. In the domain of graphs, most libraries support force-directed graph and/or treemap. The other idioms are not supported. In the domain of maps, Choropleth and symbol maps are commonly supported. Three libraries are specifically designed to visualize word cloud.

We find that most chart libraries do not accommodate the interactions brush, filter and transition. For map libraries, zoom and pan are usually equipped, but the developer needs to imperatively translate original data into visual scales, and implement other interactions. The majority of the map libraries supports drawing shapes of geographic regions based on given coordinates. Some also support tile-drawing from third-party providers to enrich the map visualizations, such as Leaflet, MapBox and OpenLayers3. However, most map libraries require low-to-mid level coding for data parsing, mapping and event handling. Developers also have to manage different map layers explicitly. Most graph-related libraries assist developers in drawing force-directed graphs and/or treemaps on a high level. Node sizes and edge weights are often automatically scaled according to input data. But balloons/tool-tips, edge arrows and edge labels are often not directly supported. Zoom and pan are sometimes supported, which is especially useful when the graph is large. For word cloud visualization, the current idiom is to show a static layout of words, of which the sizes are scaled. This can be improved by allowing user interactions to discover linkages between words and phrases. However, this requires efforts on the algorithmic front, which is beyond the scope of our inquiry.

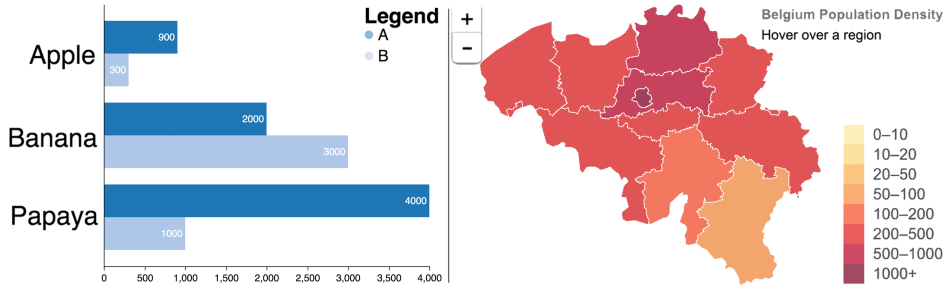

Figure 9.3: A Grouped Bar Chart (left) with Artificial Data and A Choropleth Map (right) Showing the Province-wise Population Density in Belgium

9.5.2 Case Studies with Code Snippets

We examine the libraries in more detail by investigating code snippets. We select the snippets that represent distinct API types for idiom creation, in terms of abstraction levels and programming styles.

```
1 <div class="dviz-content">
2 <pre><code>
3 Category, A, B
4 Apple, 900, 300
5 Banana, 2000, 3000
6 Papaya, 4000, 1000
7 </code></pre>
8 <p><code>(@bar)</code></p>
```

Figure 9.4: DViz, code snippet for the grouped bar chart in Figure 9.3: high abstraction with complete encapsulation

Figure 9.4, 9.5, 9.6 and 9.7 show four snippets for drawing the grouped bar chart[22] in Figure 9.3 (left). DViz (Figure 9.4) aims to maximally simplify the process of creating visualization idioms: only the data needs to be defined in a traditional tabular format (line 2-7). The library will then automatically recognize the structure of the data and implement all parts of the visualization, including visual encodings and user interactions, in this case: layout of bars, axes and legend, color coding, mouse-hover highlight with a balloon box, and legend filtering. This is the most convenient API type for developers to create visualization idioms programatically. However, the advantage is also

[22]Libraries have different color palettes and layout algorithms, the final looks of the same visualization may vary slightly.

the drawback in the sense that it lacks customizability. ChartKick (Figure 9.5) adopts a similar approach as DViz, but allows more customization by exposing configuration options. Data needs to be first transformed into JSON object, ChartKick then creates the visualization with the "BarChart" template, default parameters can be modified in JSON, as in line 10-13.

Libraries like Vega (Figure 9.7) and AmCharts (Figure 9.6) do not offer (comprehensive) idiom templates. Instead, the developer needs to explicitly define the idiom components. Vega[23] provides the interfaces for abstract components that could be applied to all visualizations: "data" (line 5-10), scales (line 11-23), axes (line 24-28), legends (line 29-31) and marks (32-64). All the specifications of a visualization are defined in JSON. This promotes visualization standardization and reusability. However, it is more verbose and difficult to read than the other libraries for idiom visualization. And it tends to be more so as the complexity of an idiom grows, because it adopts nested JSON to express visualization specifications. For instance, in Figure 9.7, the bar groups are defined as nested marks in line 32-64. AmCharts offers a "meta-template" (line 12), within which, components such as data provider (line 13-14), marks (i.e. bars in this case, line 16-29) and legend (line 30-32) need to be explicitly defined in JS.

```
1  <div id="vis"></div>
2  <script>
3  new Chartkick.BarChart("vis",
4      [ { name: "Apple",
5          data: [["A",900],["B",300]]},
6        { name: "Banana",
7          data: [["A",2000], ["B",3000]]},
8        { name: "Papaya",
9          data: [["A",4000], ["B",1000]]}],
10       {"library":
11         {"legend": {"position":"top"},
12          "hAxis": {"title":"value"},
13          "vAxis": {"title":"type"}}});
14 </script>
```

Figure 9.5: ChartKick, code snippet for the grouped bar chart in Figure 9.3: high abstraction with idiom template and configuration options in JSON

Furthermore, we find that, the data structure taken as input is dependent on specific libraries. For example, to build the grouped bar chart in Figure 9.3

[23] Vega was conceived based on d3, as a declarative language that hides the imperative details of d3, while focusing on reusable visualization design. This approach sacrifices expressiveness to some extent, and so far has been able to create static visualizations. However, vega is being actively developed, and extensions for declarative interaction design have been proposed[144]. Vega has a self-implemented scene graph, partially to accommodate canvas-rendering situations.

```
1  <div id="vis">
2  <script>
3  var chart;
4  var chartData =
5     [{"category": "Apple",
6       "A": 900,"B": 300},
7      {"category": "Banana",
8       "A": 2000,"B": 3000},
9      {"category": "Papaya",
10      "A": 4000,"B": 1000}];
11 AmCharts.ready(function () {
12     chart = new AmCharts.AmSerialChart();
13     chart.dataProvider = chartData;
14     chart.categoryField = "category";
15     chart.rotate = true;
16     // bars A
17     var graphA = new AmCharts.AmGraph();
18     graphA.type = "column";
19     graphA.valueField = "A";
20     graph1.lineAlpha = 0;
21     graphB.fillColors = "SteelBlue";
22     chart.addGraph(graphA);
23     // bars B
24     var graphB = new AmCharts.AmGraph();
25     graphB.type = "column";
26     graphB.valueField = "B";
27     graphB.lineAlpha = 0;
28     graphB.fillColors = "LightSteelBlue";
29     chart.addGraph(graphB);
30     // legend
31     var legend = new AmCharts.AmLegend();
32     chart.addLegend(legend);
33     // write
34     chart.write("chartdiv");
35 });
36 </script>
```

Figure 9.6: AmCharts, code snippet for the grouped bar chart in Figure 9.3: high-mid abstraction with idiom components in JS objects

(left), DViz takes CSV (Figure 9.4, line 3-6), ChartKick takes nested JSON array (Figure 9.5, line 4-9), AmCharts (Figure 9.6, line 5-10) takes yet another form of JSON array. This type of diversity creates barriers between data and visualization, and makes a library harder to learn. It is advisable for a library to be "standard-sensitive", taking conventional data formats, such as TSV, CSV, and transforming the data under the hood.

Figure 9.8 and 9.9 show two snippets for drawing the choropleth map in Figure 9.3 (right). In Google Charts (Figure 9.8), the choropleth or symbol map idiom is given the package name "geochart" (line 3). The developer just needs to define the data (line 5-9) to achieve the visualization, with automatic map rendering/positioning/scaling, color coding, legend drawing, and mouse-hover interactions. Appearance customization can be modified within "options" (line 10). However, the developer needs to explicitly define zoom and pan interactions. The data used in Figure 9.8, line 5-9 is the province-wise population density in Belgium.[24]

To give developers more freedom for plotting geographic regions that are not readily available, many libraries like Leaflet offer custom vector layers, on which developers can use their own shape files, e.g. SVG, geoJSON, topoJSON, etc. As shown in Figure 9.9, line 8-12, geoJSON objects, in this case, Belgian province shapes, are automatically recognized and event handlers can be attached to each

[24]Province names are ISO-3166 encoded. Also note that we use this data only for illustration purpose. However, Google Charts currently do not support Belgium map at the province resolution.

```
<div id="vis" class="view">
<script>
var spec = {
  "width": 600, "height": 400,
  "data":
  [{"name": "table",
    "values": [
      {"cat":"Apple","pos":'A',"val":900},
      {"cat":"Banana","pos":'A',"val":2000},
      ...]}],
  "scales":
  [{"name": "cat_scale",
    "type": "ordinal",
    "range": "height",
    "domain": {"data":"table",
               "field":"data.cat"}},
   {"name": "val_scale",
    "range": "width",
    "domain": {"data":"table",
               "field":"data.val"}},
   {"name": "color_scale",
    "type": "ordinal",
    "range": "category20"}],
  "axes":
  [{"type": "y",
    "scale": "cat_scale"},
   {"type": "x",
    "scale": "val_scale",}],
  "legends":
  [{"fill": "color_scale",
    "title": "fruits"}],
"marks":
[{"type": "group",
  "from": {
    "data": "table",
    "transform": [{"type":"facet",
                   "keys":["data.cat"]}]},
    "scales": [
      {"name": "pos_scale",
       "type": "ordinal",
       "range": "height",
       "domain": {"field": "data.pos"}}],
    "marks": [
      {"type": "rect",
       "properties": {
          "enter": {
            "y":{"scale": "pos_scale",
                 "field": "data.pos"},
            "height":{"scale": "pos_scale",
                      "band": true},
            "x":{"scale": "val_scale",
                 "field": "data.val"},
            "x2":{"scale": "val_scale",
                  "value": 0},
            "fill":{"scale": "color",
                    "field": "data.pos"}
          },
          "update": {
            "fill":{"scale":"color",
                    "field":"data.pos"}
          },
          "hover": {
            "stroke": {"value": "red"}
          }}}]}];
vg.parse.spec(spec, function(chart) {
  chart({el:"#vis"}).update();});
</script>
```

Figure 9.7: Vega, code snippet for the grouped bar chart in Figure 9.3: mid-low abstraction with idiom components in JSON

of these objects. However, Leaflet, like many of its peers, falls back to lower-level programming choices: manually mapping data items to colors, trivially handling events (line 13-16), and drawing legends, balloons (line 17-22). In comparison, Google Charts provides a "cleaner" event handling interface (Figure 9.9, line 16-18).

We also learn that, for map visualizations, having a server-side map provider is beneficial. Because on the one hand, custom shape files are difficult to craft and/or collect, map tiles take large amount of space to store, and when using such files, developers need to make an extra effort in preprocessing them, as the keys/ids in these files tend be diverse in format; on the other hand, a server-side map provider can store and aggregate map resources to provide intelligent/intuitive APIs. For example, a developer can use a list of free-form province names, the map provider then identifies the correct geographic shapes and returns them. Furthermore, features such as location names, administrative divisions can be standardized and updated.

```
<div id="vis">
<script type="text/javascript">
  google.load("visualization","1",{packages:["geochart"]})
  google.setOnLoadCallback(drawRegionsMap);
  function drawRegionsMap() {
    var data = google.visualization.arrayToDataTable([
      ['province','density'],['BE-WLX', 61],
      ['BE-WLG', 280],['BE-WHT', 347],['BE-VLI', 350],
      ['BE-WBR', 350],['BE-VWV', 370],['BE-VOV', 485],
      ['BE-VBR', 516],['BE-VAN', 617],['BE-WNA', 130]]);
    var options = {
      "region": "BE", "resolution": "provinces"
    };
    var chart = new google.visualization
                    .GeoChart(document.getElementById('vis'));
    google.visualization.events.addListener(chart,
      'regionClick', function(e){...}
    });
    chart.draw(data, options);
  }
</script>
```

Figure 9.8: Google Charts, code snippet for the choropleth map in Figure 9.3: high-mid abstraction with configuration options in JS objects

9.6 Mid-level Features: A Bridge

Information Visualization is much more than just conventional charts, graphs and maps. The large design space based on marks and events (Table 9.3) allows novel and more complex visualizations to be developed. For example, traditional force-directed layout becomes unreadable when the graph is relatively large, as nodes and edges overlap and occlude each other. To address this issue, developers can build visualizations that collapse or expand nodes/edges upon user interaction. The sizes of the collapsed nodes and edges are scaled according to their respective numbers of child nodes/edges, as shown in Figure 9.10.[25]

Low level instructions or high level idiom templates do not help in situations like this, including convex hull (light blue hull in Figure 9.10) creation, and data-to-visual mapping, e.g. converting data values to circle radii nonlinearly. Even if a graph-idiom library incorporate these features, it becomes unmanageable when the visualization's complexity grows. For instance, when the graph edges need to be removed or hidden from sight, and the nodes to be repositioned according to another non-graph layout. Different libraries "speak" different languages. Most of the time, operators and operands in one library are of little use to another library. This is especially the case for Canvas-based libraries that draw library-specific graphics onto a single DOM element. Therefore, unless an idiom

[25] From `http://bl.ocks.org/GerHobbelt/3071239`

```
<div id="vis">
<script>
  $.getJSON('path/to/geojson', function(provinces){
    var map = L.map('map').setView([50.5, 4.35], 8);
    L.tileLayer(
      'https://{s}.tiles.mapbox.com/v3/{id}/{z}/{x}/{y}.png',
      { maxZoom: 18, id: 'map_id'}).addTo(map);
      // encode geoJSON to visuals
      L.geoJson(provinces, {
        style: function(feature) {...},
        onEachFeature: onEachFeature
      }).addTo(map);
      function onEachFeature(feature, layer) {
        layer.on({ mouseover: function(e) {...},
                   mouseout: function(e) {...},
                   click: function(e) {...}});}
      // legend and balloon
      L.control({
        position: 'bottomright',
        onAdd: function (map) {...}
        update: function (props) {...}
      }).addTo(map);
  });
</script>
```

Figure 9.9: Leaflet, code snippet for the choropleth map in Figure 9.3: mid-low abstraction with configuration options in JS objects

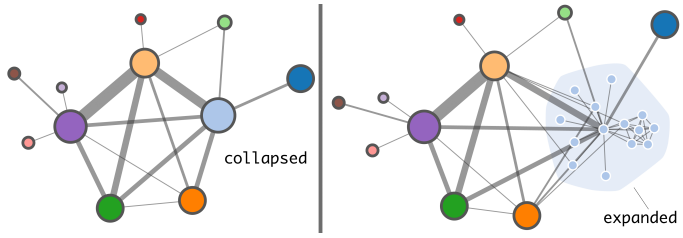

Figure 9.10: A Modified Force-directed Layout that Allows Node Groups to be Collapsed (left) or Expanded (right) upon Mouse Click

library grows to be a full-fledged general-purpose one, its use in building novel, complex visualizations is limited.

Because developing visualizations with low level instructions and utilities is inefficient, and developing novel visualizations with highly abstract, specialized libraries is restrictive or impossible, in this section, we examine the mid-level library features that lift developers from trivial, repetitive coding while maintaining expressivity. We examine these features through two lenses: visualization tasks and programming styles.

Table 9.7: Visualization Tasks

encode	visual encoding with shape, size, position, tilt/angle and color
manipulate	select, navigate, arrange, change, filter, aggregate
introduce	annotate, import, derive, export

9.6.1 Towards Visualization Tasks

In order to build novel visualizations, a library should be capable of accommodating the visualization tasks that are generally required for any type of visualization. A visualization task has two notions: from a user's perspective, it refers to the interaction task that the user performs to comprehend, explore and extract knowledge from the visualization; from a developer's perspective, it refers to the programming task to fulfil a certain requirement of the visualization. In the rest of this subsection, we first review and summarize state-of-the-art taxonomies of visualization tasks, then we translate the task requirements to library feature requirements, and examine to which extent the general libraries implement these features.

Task Taxonomies

We adopt the taxonomy in [31], which is summarized based on comprehensive literature surveys, and also covers those in [154, 92]. The tasks fall in three categories: "encode", "manipulate" and "introduce", as listed in Table 9.7.

"Encode" refers to the tasks of visually encoding data with the five aspects of marks (Table 9.3, excluding transition).

"Manipulate" refers to the user interactions that alter visual encodings to achieve data exploration or verification. In these interactions, *select* refers to the actions that differentiate selected visual elements from the unselected, such as mouse hover; *navigate* refers to the actions that change the user's view point, such as zoom; *arrange* refers to actions that organize visual elements spatially, such as layouts; *change* refers to the transitions between different visual encodings; *filter* refers to the actions that adjust the exclusion or inclusion of visual elements; *aggregate* refers to the methods that change the granularity of visual

Table 9.8: Middle-Level Features for General Visualization Tasks

Visualization Tasks		Mid-level Library Features
encode	*parts*	axes, legends, balloons, layouts
	utilities	array-util, geom-util, scales, formatters, locale
manipulate	*select*	brush, lasso, point-selection
	filter	subset-query, cross-filter, free-query
	navigate	zoom, pan, collapse/expansion, history
	arrange	collision/intersection, view-layouts
	change	interpolators, data-update
introduce	*annotate*	create, copy, remove, modify, free-drawing
	import	data-import, visual-import
	export	data-export, visual-export
	derive	math, analytics

elements, such as the node expanding/collapsing in Figure 9.10; *guide* refers to the interactions for a user to become familiarized with a visualization.[26]

"Introduce" refers to the user interactions that add or export new graphics for the documenting and sharing a visualization. In these interactions, *annotate* refers to the addition of graphical or textual annotations associated with visualization elements; *import* refers to the data import process to a visualization; *export* refers to the derivation of new data elements based on existing ones; *record* refers to the methods that save or capture visualization elements as persistent artefacts.

From Tasks To Features

Each of these tasks requires a set of programming routines that one encounters frequently and repetitively when developing visualizations. The mid-level library features help developers (partially) avoid these routines. We translate the tasks to essential features in Table 9.8. Note that the feature catalog is non-exhaustive, as there could be potentially more or different library features that fulfil the

[26] We add the visualization task *guide* from [92] to this taxonomy for completeness.

requirements of the visualization tasks. We compose this catalog based on our own experience in data-visualization development.

To visually encode data, there are some common parts, including axes and legends, as we have seen from Section 9.5, they are almost always the necessary components for charts or maps. Another common part is balloons, or pop-ups, which are the little boxes on the top/side of visual elements to show more information. Layouts are positional pre-configurations for various idiom visualizations. But different from the high-level approach, the layouts are decoupled from other encoding factors to allow more flexible customization, for example, a switch from the force-directed layout in Figure 9.10 to a bubble-chart layout. Also, various utilities are necessary to mitigate the burden of encoding calculation. Array utilities (*array-util*) are essential to data visualization, as array is the primary structure to create, update and remove visual elements. There is a wide range of array utility functions, common ones include: `min()`, `max()`, `mean()`, `sort()`, `shuffle()`, `hashmap()`, etc. Geometry utilities (*geom-util*) contain a wide range of functions that help developers construct geometric shapes or calculate geometric properties. The implementation of these features depends on specific libraries. Common features include computation of the bounding box or convex hull of given visual object(s), computation of the centroid of given visual object(s), graticule generation, geographic coordinate projection, etc. *Scales* are at the core of general libraries, they are the mapper functions that convert data values to visual ranges. *Formatters* are the functions that automatically parse, manipulate and/or convert data formats, for example, parsing a string to an ISO time format or computing the number of days between two given two dates. Localization (*locale*) is also an important part of visualization libraries that makes visualizations accessible to a greater audience.

To make visualization manipulatable, libraries need to accommodate a series of user interactions. In terms of selection, *brush* is the mouse-dragging action that draws a rectangle to define an area on the screen. It is often used in combination with the *cross-filter* feature to enable "focus and context" exploration[16]. *Lasso* is an alternative of *brush* in free form, the user "lassos" around visual objects to make a selection. For both "brush" and "lasso", a library needs to recognize the user's mouse-dragging action and identifies the enclosed visual objects. *point-selection* refers to the underlying computation feature that detects and selects the positional points defined by user's current mouse/touch trace or location. This feature is often used in combination with *brush* or *lasso* to select an area of points, or to support ambiguous point selection. In terms of filtering, *subset-query* refers to the underlying filtering mechanism that takes formulated expressions (e.g. mathematical comparisons, regular expressions) and returns the matched data subset. *Cross-filter* is the data utility that manages the filtering of data instances across multiple attributes simultaneously.

Free-query is the advanced version of *subset-query*, it takes a freely formulated query, often in words, and returns the matched subset. A common usage of *free-query* is the search box where the user types in texts to filter a visualization. In terms of navigation, *zoom* and *pan* are the most common techniques for navigating through a visualization. The corresponding features need to manage coordinate re-mapping and shifting under the hood. *Collapse/expansion* is a set of techniques that allows the user to "summarize" and "inspect in detail" in a large-scale visualization, we have seen the typical example in Figure 9.10. The corresponding library feature manages the re-calculation of layout upon "collapse/expansion". *History* is the feature that keeps track of user actions, allowing "step-back" and "step-forward", etc. It is often needed when a user interacts with a visualization and wants to go back to the previous state. However, this requires a library to be able to describe and store user actions. In terms of (re)arranging visual elements, *collision detection* and *intersection* calculation are also needed to allow a visualization to "self-correct", preventing occlusions. For example, when a user add more lines to a line chart, the text labels on the sides the lines tend to overlap. Collision-detection help alleviate this problem. *View-layouts* are the layouts for multiple visualizations, typically useful for "dashboard-like" applications and cross-filtering, but often trivial and repetitive to implement. In terms of dynamically changing the states of visual objects, *interpolators* are essential for shape drawing and transition rendering. They help developers focus on defining the key, discrete states of visual objects, and leave the rest to interpolation. *Data-update* refers to the feature that detects underlying data change, identifies the visual objects that need be created, updated or moved. This feature is particularly useful when visualizing stream data. We omit the "aggregate" category for the "navigate" category has covered the *collapse/expansion* aspect.

Users also often need to create their own visualizations, annotate them, deduce new knowledge, and share them with others. In terms of annotation, *create*, *copy*, *remove* and *modify* refer to a series of actions that a user takes to annotate a visualization. Any visual object, including its components and properties, is subject to these actions. For example, a user thinks the point labels in a line chart are not suitable, and wants to directly modify them in the visualization. *free-drawing* allows a user to draw annotation in free form, such as hand drawing an arrow pointing to a node. There could be more sophisticated annotating actions, as we often see in specialized text or graphics editing applications. Here we examine the basic ones. In terms of import, libraries should provide facilities to parse imported data in different formats (*data-import*), as covered in Table 9.6 from the previous section. Furthermore, visual specification import (*visual-import*) makes it possible for a visualization to directly adopt alternative visual properties without refactoring the source code. However, this requires libraries to "understand" standardized visual languages. In terms of export,

Table 9.9: Coverage of Middle-Level Features by the General Libraries (excluding d3)

	encode				**manipulate**		**introduce**	
library	*array-util*	*geom-util*	*scales*	*formatters*	*interpolators*	*intersection*	*free-drawing*	*data/visual-import*
easel	×	b-box	×	×	easing func.	×	×	×
fabric	×	b-box	×	×	easing func.	×	free-brush	×
kinetic	×	b-box	×	×	easing func.	×	×	×
paper	×	b-box	×	×	×	✓	×	JSON, SVG, image
processing	sort, reverse, hashmap, etc.	×	linear	string, time, number	×	×	×	SVG, XML, image
raphael	×	b-box	×	×	easing func.	✓	×	×

sometimes a user needs to share a visualization in other formats, such as .png, .jpg images, SVG definitions, or a user wants to download the dataset behind a visualization, either original or modified due to later interactions, *visual-export* and *data-export* facilities address these requirements. Note that the *annotate*, *import* and *export* features also contains respective user interfaces to accommodate the corresponding actions, for example, an "import data" button that lets a user choose a local dataset, or a drop-down menu that lets a user choose an image format to export. Finally, data visualization can go beyond visualizing relatively simple, raw data. It can leverage the advancements in the fields of data mining and machine learning, so as to infer knowledge from given data, then visualize the knowledge. A library equipped with such features has richer data input, and can produce more sophisticated visualizations. The *math* and *analytics* features in the "derive" category refer to these features. A few basic functions in these features are also covered in *array-util*.

Feature Coverage

All the general libraries, except d3, are designed to accommodate predominately low-level drawing. We first summarize the features that are covered to some extent by the six libraries (excluding d3) in Table 9.9, then inspect d3 in more detail.

From Table 9.9, we can see that among six libraries, array utilities are seldom supported, processingJS provides some basic ones, and further offer convenient array structures such as HashMap, ArrayList. In geometry utilities, only bounding box (b-box) is available, which calculates the rectangle enclosing a given visual object. For data scaling, only processingJS provides a linear scaling function `map()` that linearly maps a data value to visual domain. For data formatting, processingJS provides a few basic formatters for string, time

and number. For visualization manipulation, some libraries provide a few easing functions for transitions. paper and raphael provide utilities to calculate intersections, such as detecting whether a point is inside an area, deriving the intersecting point between two lines/curves. Moreover, fabric has a few brush utilities for free drawing (different from the "brush" selection). paper and processingJS provide utilities to load files with different formats, such as JSON, SVG. d3 is more comprehensive than the other six. Its strong points are the following:

- *encoding-parts*: axes and a rich set of layouts including idiom layouts and others
- *encoding-utilities*: various array and geometry utility functions, including a range of geographic projections; a rich set of scales that convert between discrete and continuous values (non)linearly; a range of formatting utilities for number, string, and time
- *select*: brush and point-selection utilities, including Voronoi tessellation and quad-tree, which are space division strategies for efficient point search.
- *change*: a rich set of interpolators that facilitate path-drawing, data-scaling/converting and transitions; a mechanism for data binding and update, which automatically identifies the visual elements corresponding to the incoming and outdated data, so that developers can create new visuals, update existing ones and remove obsolete ones without explicit data-binding
- *data-import*: CSV, TSV, HTML, JSON, XML, text, HTTP request, etc.

In general, the features listed in Table 9.8 are fairly well supported by d3. But there are a few unaddressed features, which are also the common weakness of all general libraries: *legends*, *balloons*, *lasso*, *cross-filter*, *free-query*, *collapse/expansion*, *history*, *collision/intersection* and most features in the "introduce" category. The features for annotating and exporting a visualization, and data analysis can be accommodated straightforwardly by third-party libraries, as these features are decoupled from a visualization's internal encoding and manipulating mechanism.[27]

Furthermore, though zoom and pan behavior in d3 is supported so that the developer needs not explicitly code positional translation and scaling, there are flaws: semantic zooming [63] can conflict with its layouts (e.g. force-directed),

[27]The exception is exporting vector graphics from Canvas-based visualizations, since the definitions of visual objects are library dependant.

```
{ "groups":[
    {"id":1,
     "children":[
        {"name":"Fantine"},{"name":"Valjean"}
     ]},
    {"id":2},...
  ],
  "group-links":[
    {"source":1,"target":2,"w":1},
    {"source":2,"target":3,"w":8},...
  ],
  "node-links":[
    {"source":"Valjean","target":"Fantine","w":.5}
    {"source":...}
  ]
}
```

Figure 9.11: Example JSON Dataset for the Collapsable/Expandable Force-directed Graph Visualization in Figure 9.10

and a click-event is misinterpreted as panning-start/end events. Also, for all SVG-based libraries, "transform" instructions, though convenient in global geometrical manipulation, have the disadvantage of using relative coordinates that are difficult to trace for later usage. d3 solves this issue by maintaining the absolute coordinates in the corresponding visual object's data structure. However, there is still one issue that SVG-based libraries need to address: global "translate" causes the entire background rectangle to shift. This can be undesirable for webpage layout and requires extra manual work to avoid occlusions. Another inconvenience of d3 comes from its data-binding mechanism where data items are bound to visual objects. The default binding key is the array-index of a data item. This binding can be easily corrupted when the data array changes (e.g. by the `splice()` function). A straightforward way to fix this is by injecting an "id" property to each data item if the "id" property is not explicitly defined.

9.6.2 Case Studies with Code Snippets

Besides providing useful features, it is also important for a general library to have an effective programming style. It should strive to balance between expressiveness and abstractness. We examine three representative styles: (1) the Canvas-based, with self implemented scene graph, and declarative visual creation, such as those of easel, fabric, kinetic and paper; (2) the SVG-based, with DOM as scene graph, and declarative visual creation, such as those of d3 and raphael; (3) the Canvas-based, without a scene graph, and with imperative visual creation, such as that of processingJS.

```
<div id="vis"></div>
<script>
$.getJSON("example.json", function(json){
    var colors = function(id){...};
    var rScale = function(val){...};
    var lScale = function(val){...};
    var forceLayout = function(nodes,links){...};
    var stage = new Kinetic.Stage({
        container: 'vis',
        width: 800, height: 600
    });
    var groupLayer = new Kinetic.Layer();
    for(var i=0;i<json['groups'].length;i++){
        var group = new Kinetic.Circle({
            // x,y to be determined by forceLayout
            radius:rScale(
               json['groups'][i].children.length),
            fill:colors(json['groups'][i].id,
            stroke: 'black'
        });
        group.on('click', function(){...});
        groupLayer.add(group);
    }
    stage.add(groupLayer);
    var linkLayer = new Kinetic.Layer();
    for(var i=0;i<json['groups'].length;i++){
        var link = new Kinetic.Line({
            points: [...], // determined by groups
            stroke: 'black',
            strokeWidth:lScale(json['group-links'][i].w)
        });
        linkLayer.add(link);
    }
    stage.add(linkLayer);
    forceLayout(groupLayer,linkLayer);
};
</script>
```

Figure 9.12: Kinetic, code snippet for the collapsable/expandable force-directed graph in Figure 9.10: Canvas-based, self scene graph independent of DOM, declarative programming

To illustrate these styles, we consider an example dataset in JSON, as shown in Figure 9.11, which is used to construct the collapsable/expandable force-directed graph layout in Figure 9.10. The dataset consists of groups, group-links and node-links. Each group has an id and a set of nodes as children (line 2-6). Each node has a name. Weighted links form between nodes (line 13-14), the node-links are then summarized into group-links based on the memberships of the nodes, the weights of the node-links are accumulated. Initially in the visualization, all nodes are hidden, only group-nodes are shown. The positions of the group-nodes and group-links are determined by a force-directed layout algorithm. The radii of the group nodes are scaled according to the number of child nodes from each group. Figure 9.12, 9.13 and 9.14 show the code snippets in the three paradigms that achieve the initial visualization.

```
<script>
var width = 800, height = 600;
var colors = d3.scale.category20();
var rScale = d3.scale.pow().exponent(2);
var lScale = d3.scale.pow().exponent(.5);
var svg = d3.select("body").append("svg")
.attr("width",width).attr("height",height);
d3.json("example.json", function(json) {
    var force = d3.layout.force()
        .size([width, height])
        .nodes(json['groups'])
        .links(json['group-links'])
        .start();
    var link = svg.selectAll(".link")
        .data(json['group-links'])
        .enter().append("line")
        .attr("class", "link")
        .style("stroke-width",function(d){
          return lScale(d.value);
        });
    var node = svg.selectAll(".node")
        .data(json['groups'])
        .enter().append("g")
        .attr("class", "node");
    node.append("circle")
        .attr("r", function(d){
          return rScale(d.children.length);
        })
        .style("fill", function(d) {
          return colors(d.id);
        })
        .on("click", function(){...});
    force.on("end", function() {
        link.attr("x1", function(d){return d.source.x;})
            .attr("y1", function(d){return d.source.y;})
            .attr("x2", function(d){return d.target.x;})
            .attr("y2", function(d){return d.target.y;});
        node.attr("transform", function(d) {
            return "translate(" + d.x + "," + d.y + ")";
        });
    });
});
</script>
```

Figure 9.13: d3, code snippet for the collapsable/expandable force-directed graph in Figure 9.10: SVG-based, DOM scene graph, declarative programming

In the kinetic snippet (Figure 9.12), a "stage" is first created as the scene graph root (line 8-11), later a group layer (line 12) and a link layer (line 24) are created, for each group node and group link, a circle and a line are created, attached to the respective layers. We see that via this "stage-layer" structure, basic shapes can be organized into a hierarchy, more complex visuals can be constructed and distinguished. When necessary, shapes can also be grouped with `Kinetic.Group()`, and added to a layer. In this way, developers abstract visual elements into "composites", making the code more reusable. However, the drawback of this approach is that visuals are not bound to data, namely

HTML

```
<canvas id="vis"
    data-processing-sources="sketch.pde"></canvas>
<script>
    $.getJSON("example.json", function(json){
        var pjs=Processing.getInstanceById('vis');
        pjs.init(json);
        pjs.drawLayout();
    });
</script>
```

sketch

```
//sketch.pde
List<Node> nodes=new ArrayList<Node>();
List<Edge> edges=new ArrayList<Edge>();
int colors(int id){...}
int rScale(int val){...}
int lScale(int val){...}
void forceLayout(
    List<Circle> nodes,
    List<Edge> edges){...}
void init(json) {
    size(800,600);
    for(int i=0;i<json['groups'].length;i++){
        float r =
        rScale(json['groups'][i].children.length);
        int c = colors(json['groups'][i].id);
        Node n = new Node(r);
        ...
    }
    for(int i=0;i<json['group-links'].length;i++){
        float w = lScale(json['group-links'][i].w);
        ...
    }
    forceLayout(nodes, edges);
}
void drawLayout(){
    for(int i=0;i<nodes.size();i++)
        {nodes.get(i).draw();}
    for(int i=0;i<edges.size();i++){
        {edges.get(i).draw();}
}
void mouseClicked(){
    for(int i=0;i<nodes.size();i++){
        nodes.get(i).mouseClicked();
    }
}
class Node{
    public boolean expanded;
    public int x,y,radius,color;
    public List<Node> children;
    public Node(radius){
        this.radius = radius;
    }
    public void draw(){
        fill(color);
        if(!expanded)
            ellipse(x,y,2*radius,2*radius);
        else{...};//draw children
    }
    public void mouseClicked() {
        expanded = !expanded;
        if(expanded) ...
    }
}
class Edge{
    Node source, target, weight;
    public Edge(Node source,Node target,
        float weight)
       {this.source = source;
        this.target = target;
        this.weight = weight;}
    public void draw()
       {stroke(0);
        strokeWeight(weight);
        line(source.x, source.y,
            target.x, target.y);}
}
```

Figure 9.14: ProcessingJS, code Snippets for the collapsable/expandable force-directed graph in Figure 9.10: Canvas-based, no scene graph, imperative programming

when a visual element needs to change its appearance or behavior based on the corresponding data item, the developer has to go back to the data array, or explicitly build a visual-to-data dictionary. Furthermore, kinetic, like its peers easel, fabric, etc., does not provide necessary features, in this case, the color palette that produces distinct categorical colors, the scales that map data items to visual parameters (circle radius and link width), and the force layout itself. The developer needs to implement them (line 4-7).

In the d3 snippet (Figure 9.13), the color palette, scales and force-layout are built in. We choose to use the exponential scales to emphasize group and link differences. The `svg` node is created as a DOM element, the group nodes and links are subsequently attached to it. We see a major advantage over the previous paradigm: "data binding". First, the target visual elements are collectively selected (line 14, 21), and then data-binding is automatically performed by

attaching the `data` function (line 15, 22). Each instance of a provided data array is bound to a selected visual element, and the instance can be readily retrieved in subsequent function chains (e.g. line 18, 26). A slight drawback of this approach is that, to ensure unique collective selection, elements must be attached with identifiable "class" or "id" (line 17, 24). d3 further offers a mechanism for data update management. In this mechanism, new data items that are yet to be bound to visual elements are identified and can be handled with `selection.enter()` (line 23), old visual elements that no long have corresponding data items are handled with `selection.exit()`.[28] Lastly, the positions of the links and nodes are updated when the force-layout finishes calculating (line 33-41). We can also see that d3 directly adopts the vocabulary of native SVG. It uses the `g` (short for "group") element to construct a visual composite, multiple basic shapes can be attached to `g` (line 23). For example, labels can be added to group nodes: `node.append("text")`. A change in `g` leads to the same change for the child elements. Furthermore, in order to compose more complex visual objects, the developer needs to "wrap" the constituent drawing routines into a Javascript function, and call it. A typical example is the library's built-in composite *axis* that is invoked by `selection.call(axis)`. While using a native vocabulary and scene graph promotes standards, and ease of debugging, it does steepen the library's learning curve: (1) composites need be encapsulated as functions to be called, rather than as visual components to be appended, creating a syntax inconsistency; (2) native vocabulary can be flawed. For example, a circle's position attributes are named "cx" and "cy", an path's point-array attribute is named "d", such attribute names can be counter-intuitive and make the learning curve steeper.

In the processingJS snippets, the drawing routines are programmed in a separate file in Java syntax (Figure 9.14, sketch), the functions in this sketch can then be invoked in Javascript (Figure 9.14, HTML). From the sketch, we can see that, because the library does not have built-in shape, composite, or scene graph, visual objects such as nodes and edges need to be completely coded from scratch (line 36-66). The developer also needs to code the scales and force-layout, etc. (line 4-9) and manage mouse interactions (line 31-35). While this paradigm offers expressiveness and flexibility, and can be of great value for developers familiar with traditional object-oriented programming, it lacks necessary abstractions such as those in d3: automatic data binding and update, scales, layouts and scene-graph management.

In this section, we have compared three representative programming paradigms of the general visualization libraries and demonstrated the use of mid-level features. We can see that though d3 is more advanced, it has limitations due to

[28] More detailed explanation can be found in the online tutorials `https://gist.github.com/mbostock/3808218`.

its close relationship with native technologies. By comparison, Canvas-based paradigm has the potential to be enriched with mid-level features. It can also offers an independent scene graph with a more unified and consistent visual syntax.

9.7 Future Work

In this section, we examine the current generation of visualization libraries more broadly, in terms of design patterns and smart defaults. We discuss existing solutions and propose improvement points for future work.

9.7.1 Design Patterns

As Knuth [103] said: "let us concentrate rather on explaining to human beings what we want a computer to do." Following design patterns[64] makes code more structured, readable, and thus more maintainable and reusable. We examine the design patterns presented in [64], and select those that can be found in existing visualization libraries, or those that inform future library design.

Template: The "template" pattern is common for abstracting away implementation details, we have seen it in visualization idiom libraries and the layout features. We learned that it is important to hide low-level imperative instructions in these templates, and only expose necessary configuration options, which many libraries fail to do.

Decorator: We touched the "decorator" pattern in the previous section. For example, most libraries lack the features that assist developers in building facilities that allow users to annotate or export visualizations. This can be remedied by "decorating" a visualization with ad-hoc components, such as an "annotator" or "exporter", which recognizes but does not interfere with internals of the visualization.

Composite: We have seen the "composite" pattern from the previous section, namely composites of basic shapes and/or data items can be formed via scene-graph addition and direct OO programming. The OO paradigm operates on a low-level, the scene-graph paradigm offers basic abstractions for primitive shapes. However, it does not fill the "semantic gap" between basic shape construction and more advanced composite construction. In other words, composite objects can not be directly used to form more complex composite objects. Corresponding

solutions start to appear. For example, d3.chart[29] is a library that builds an abstraction layer on top of d3. It allows chart composition in a unified syntax.

Builder: There is also the "builder" pattern that separates construction of a complex object from its representation. CSS adopts this pattern. It benefits the libraries that have native access, such as d3 and raphael. But for the other libraries, object construction and representation are still tightly coupled. Representation languages or other mechanisms need to be developed to decouple the two, if existing ones cannot be utilized.

Memento: The "memento" pattern refers to the dynamic externalization of the states of visual objects. These states include not only representations, but also behavior. They need to be described and stored so as to be referred back and forth during user interaction. It is particularly powerful when applied to user navigation through a visualization. The developer does not need to redefine every color, size, position change, etc. at each change of state. It is also useful in history manipulations such as "step forward", "step backward". This pattern is missing in the current generation of visualization libraries.

Mediator: The "mediator" pattern defines a domain that encapsulates how a set of objects interact. It promotes loose coupling by keeping objects from referring to each other explicitly, and allows developers vary the interaction independently. d3 distinguishes itself from the other libraries regarding this pattern because of its "data-driven" nature. Visual changes are constantly referred back to the corresponding data changes, which concentrates the "logic" of a visualization, and inhibits potentially messy communication between visual objects. Using data itself as the mediating domain that governs the behavior of visual objects seems to be the most natural choice for a data visualization library. However, of the libraries we review, only d3 and d3-based libraries adopt this pattern.

9.7.2 Smart Defaults

In the process of reviewing available visualization libraries, and in our own experience of developing visualization applications, we find that, for a library design, "smart defaults" are as important as design patterns and feature sets. They should be designed and implemented in the next generation of visualization libraries. By "smart defaults", we mean that the properties of visual elements are automatically given default values upon construction, without the developer specifying them. These default values should be carefully selected or calculated based on empirical evaluations on "what makes a visualization good". In

[29] `http://misoproject.com/d3-chart/`

the following, we list two aspects that "smart defaults" can contribute to visualization development, and illustrate their usefulness and necessity with examples.

Occlusion Handling: A rather frequent task that a developer encounters is to resolve visual occlusion. For example, to build a multi-line chart, the developer needs to explicitly code a process that resolves the overlapping of line labels and axis labels, as shown in Figure 9.15.[30] Each of the labels "Austin", "New York" and "San Francisco" is appended at the ending point of each line, partially occluding each other. The tick labels for the x-axis are also overlapping. Moreover, the label "San Francisco" is partially clipped by the displaying window. The process of resolving these kinds of occlusions is trivial, but takes much time. The developer needs to go with an "trial and error" approach to gradually adjust label positions, rotations, window size, etc. to remove the occlusions. Libraries should automatically avoid occlusions by calculating "smart" default values of visual properties. But the properties can still be modified later.

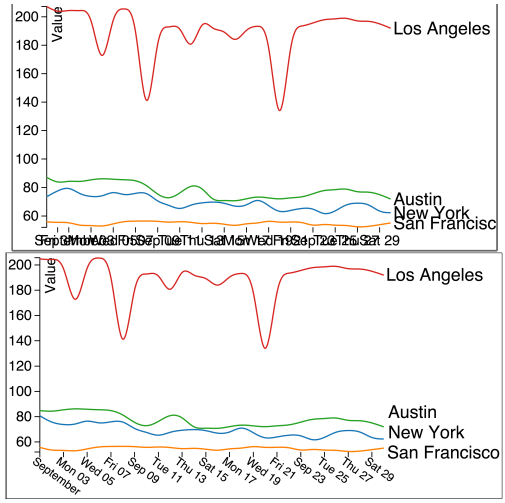

Figure 9.15: An Multi-line Chart with Occlusions (top), and with Smart Defaults that Avoid Occlusions (bottom)

Visualization Retrieval: Visualizations are produced every day to solve a variety of problems and the majority of them are open-sourced. Much of the time, when a developer wants to build a visualization towards a certain problem, there has already been a visualization coded and shared on the web that meets

[30] reused example from `http://bl.ocks.org/mbostock/3884955`

the requirements. For example, a developer wants to build a multi-line chart along with a world map to synchronously show different dimensions of a dataset. There might be an example visualization that meets these criteria to some extent. We consider this type of resource also as a form of defaults, which provides developers with immediately reusable and learnable snippets of codes, along with the corresponding visualizations. What is much needed is a platform that aggregates and maintains these resources, offers developers interfaces to search, rank and retrieve visualizations and codes.

9.8 Conclusion

With the technological advancements in the areas of browser-based graphics and Javascript engines, more sophisticated interactive data visualizations can be developed as part of the web application ecosystem. We have seen many libraries for online visualization come into play. Some of these libraries are widely used by visualizations developers, some libraries are studied and/or redesigned by developers for alternative or custom use. The developers and/or library designers in both industry and academia rely on these libraries heavily to build their own visualization systems, be it for data exploration and analysis, or for user studies. However, so far, there is a lack of a comprehensive survey of the current library landscape, a systematic comparison and evaluation of the libraries and proposals for future improvements. In this work, we fill in the gap.

More specifically, we reviewed over 100 Javascript data visualization libraries and summarize their application domains, technologies and abstraction levels (Section 9.3). We then categorized and examined the libraries on different levels of abstraction and found that though there was abundant libraries available, most targeted a very limited range of problems on a high abstraction level. We first composed a taxonomy of low-level instructions and utilities for creating custom visualizations according to the literature on basic visualization components (Section 9.4). The general libraries were then compared and evaluated against this taxonomy. Next, we studied the high-level libraries (Section 9.5) based on a comprehensive catalog of visualization idioms. We found that only common charts and maps were widely supported. But graph idioms were under-supported. We then developed a taxonomy for the evaluation of idiom libraries, and identified improvement points. For example, most chart libraries did not accommodate brush and filter interactions. Developers can also utilize Table 9.10 and 9.11 to select the idiom libraries they need. We further discussed different coding styles, identified merits and drawbacks of the idiom libraries by discussing code snippets.

To create novel visualizations, developers need to be freed from basic drawing instructions and enabled to concentrate on a higher level of abstraction, without having to sacrifice expressiveness. We composed another taxonomy for evaluating the capabilities of general libraries by translating visualization task requirements to mid-level features (Section 9.6). We examined these features in detail and discussed the merits and drawbacks of different programming styles of the general libraries. Finally, we proposed improvements for the current generation of visualization libraries in terms of design patterns and smart defaults.

In summary, we developed a comprehensive approach to compare and evaluate Javascript libraries for interactive data visualization. Via this approach, visualization developers can strategically choose the library they need. Visualization library developers can improve or extend existing libraries, or design the next generation of libraries with more informed choices and a more systematic methodology.

Table 9.10: Summary of the Javascript Data-Visualization Libraries. Part I

Name	Domain	Idioms	Lack	Renderer	Lic.	Website	Update
amchart	chart, map	*		SVG	both	http://goo.gl/wb5pC2	2015
ApertureJS (raphael, OpenLayers3)	chart, map, graph	bar, line, pie	low-mid	Canvas, SVG, WebGL	free	http://aperturejs.com	2015
arbor.js	graph	force	low-mid	Canvas, SVG	free	http://arborjs.org	2012
AwesomeChartJS	chart	bar, line, pie, ring	no interact.	Canvas	free	https://goo.gl/WkjeOR	2012
Bluff	chart	line	no brush	Canvas	free	http://goo.gl/SLXOu8	2010
bonsai	general			SVG	free	http://bonsaijs.org/	2015
C3 (d3)	chart	*	no brush	SVG	free	http://c3js.org	2015
Canvas3DGraph	3D			Canvas	free	http://goo.gl/lzkDQ5	2007
CanvasJS	chart	* - radar	no brush, filter	Canvas	both	http://canvasjs.com	2013
CanvasXpress	chart, graph, text	* - radar, ring	no brush, filter	Canvas	free	http://goo.gl/Oudp6j	2013
Cartographer.js (Google map)	map	chorop., symbol	no interact.	Canvas, SVG, WebGL	free	http://goo.gl/VvwNT4	2010
ccchart	chart	* - pie, radar	no interact.	Canvas	free	http://ccchart.com	2015
chart.js	chart	* - bubble	no brush, filter	Canvas	free	http://www.chartjs.org	2015
chartkick.js	chart, map	* - radar, bubble	no brush, filter	SVG	free	https://goo.gl/Wuqdxw	2015
Chroma.js (ColorBrewer)	color			Canvas, SVG	free	http://goo.gl/w0BipJ	2015
Chromatist	color			Canvas, SVG	free	https://goo.gl/Bsvu9Z	2012
cola.js (d3)	graph	force	low-mid	SVG	free	http://goo.gl/0uQlZi	2015
ColorBrewer	color			Canvas, SVG	free	http://colorbrewer2.org	?
Cubism.js (d3)	chart	line	no brush, filter	SVG	free	http://goo.gl/sDG49v	2015
Cytoscape.js	graph	force, rect. node-link	low-mid	SVG	free	http://goo.gl/MCyht6/	2015
d3	general			SVG	free	http://d3js.org/	2015
d3cloud (d3)	text	word cloud		SVG	free	http://goo.gl/D6nXni	2015
d3geomap (d3)	map	chorop., symbol		SVG	free	http://goo.gl/RrXf3s	2015
d3pie (d3)	chart	pie, ring	no filter	SVG	free	http://goo.gl/eqMGyO	2015
d3parcoords (d3)	chart	par.coord.		SVG	free	https://goo.gl/Lp8CyD	2015
d3plus (d3)	chart, map, graph	* - ring, radar		SVG	free	http://d3plus.org/	2015
Datamap (d3)	map	chorop., symbol	low-mid	SVG	free	https://goo.gl/Dr3J3W	2015
dc.js (d3)	chart	* - ring, radar		SVG	free	http://goo.gl/w3Ti7f	2015
dHTMLxChart	chart	*	no interact.	SVG	both	http://dhtmlx.com	2015
Diagram Builder	chart, graph	bar, line, pie	low-mid	DOM	free	http://goo.gl/9Sk33s	2007
dimple (d3)	chart	* - radar	no brush, filter	SVG	free	http://dimplejs.org	2015
Dojo (OpenLayers 3)	chart, map	(* - radar) (chorop., symbol)	low-mid	Canvas, SVG, WebGL	free	http://dojotoolkit.org	2015
dviz (d3, Google chart)	chart, graph	(* - pie, radar) (force)		SVG	free	https://goo.gl/q4KQen	2012
dygraph	chart	line	no filter	Canvas	free	http://dygraphs.com	2015
easel	general			Canvas	free	http://createjs.com/	2015
Elychart (raphael)	chart	* - bubble	low-mid	SVG	free	http://elycharts.com	2015
Ember chart (d3)	chart	* - ring, radar	no brush, filter	SVG	free	http://goo.gl/jKPQM7	2015
Ember Timetree (d3)	chart	bar	no brush, filter	SVG	free	http://goo.gl/Oy59e9	2013
Envision.js (Flotr2)	chart	line	low-mid	Canvas	free	http://goo.gl/Ekpmyy	2013
fabric	general			Canvas	free	http://fabricjs.com/	2015
Flot	chart	* - radar, ring	low-mid	Canvas	free	http://goo.gl/5GvE5e	2015
Flotr2	chart	*		Canvas	free	http://goo.gl/X2g7Y9	2015
Fusionchart	chart, map			SVG	comm.	http://goo.gl/D3oKSn	2015
gmap.js (Google map)	map	chorop., symbol		Canvas, SVG, WebGL	free	https://goo.gl/mDSRuJ	2015
Google chart	chart, map	*		SVG	free	https://goo.gl/OCcfam	2015
Google map	map	chorop., flow symbol		Canvas, SVG WebGL	free	https://goo.gl/zVmDux	2015
Grafico (raphael, Protovis)	chart	* - bubble, pie, radar, ring	no brush, filter	SVG	free	http://goo.gl/zxOtci	2011
graphael (raphael)	chart	* - radar, ring	low-mid	SVG	free	http://goo.gl/RJxcRB	2012
graphene (d3)	chart	line		SVG	free	http://goo.gl/lJOlYn	2015
Harry Plotter	chart	* - radar, bubble	no brush, filter	Canvas	free	http://goo.gl/NkiLzF	2015

Table 9.11: Summary of the Javascript Data-Visualization Libraries. Part II

Name	Domain	Idioms	Lack	Renderer	Lic.	Website	Update
heatmap.js	chart	heatmap		Canvas	free	http://goo.gl/Sv8EKt	2015
Highchart	chart, map, graph			SVG	free	http://goo.gl/CjAgDB	2015
HTML5 wordcloud	text	word cloud		Canvas	free	http://goo.gl/MjdFHT	2015
Ico (raphael)	chart	* - pie, radar	no interact.	SVG	free	http://goo.gl/UzRbdK	2012
JointJS	chart, graph	line, pie ring, node-link		SVG	free	http://jointjs.com	2015
jqPlot	chart	* - radar	no brush, filter no highlight	Canvas	free	http://www.jqplot.com	2015
jQuery.sparklines	chart	* - radar, ring	no brush, filter	Canvas	free	http://goo.gl/Gl6ynj	2013
jQuery.orgchart	graph	node-link	no interact.	DOM	free	https://goo.gl/5n3K2k	2015
jQuery.spidergraph	chart	radar	no interact.	SVG	free	https://goo.gl/OePuF5	2013
JS Infovis Toolkit	chart, graph	(* - line, radar) (force, treemap, node-link)	no brush, filter	Canvas	free	http://goo.gl/tiFhnE	2015
JSchart	chart	* - radar	no brush, filter	Canvas	both	http://ejschart.com/	2013
JSXGraph	comput.plot.			Canvas, SVG	free	http://goo.gl/BPa3jT	2015
Kartograph.js (raphael)	map	chorop., symbol		SVG	free	http://kartograph.org/	2015
Keylines	graph			Canvas	comm.	http://keylines.com	2015
kinetic	general			Canvas	free	http://kineticjs.com	2015
KoolChart	chart			Canvas	comm.	http://goo.gl/to0zFR	2015
Leaflet	map	chorop., flow symbol	low-mid	SVG	free	http://leafletjs.com/	2015
Mapbox (Leaflet)	map	chorop., flow symbol	low-mid	SVG	free	https://goo.gl/r0Ri6S	2015
modestmap	map	chorop, flow symbol	low-mid	SVG	free	http://goo.gl/HXo0zj	2015
Morris.js (raphael)	chart	* - pie, bubble	no brush, filter	SVG	free	http://goo.gl/5HympH	2015
mxGraph	graph			Canvas, SVG	comm.	http://goo.gl/Fcrq75	2015
NVd3 (d3)	chart	*	no brush	SVG	free	http://goo.gl/NSt4LZ	2015
OLAPchart	chart	*	low-mid	Canvas	free	http://goo.gl/LRhNuv	2015
OpenLayers 3	map	chorop., symbol	low-mid	Canvas, WebGL	free	http://goo.gl/aXh2D5	2015
paper	general			Canvas	free	http://goo.gl/xlNFNS	2015
Peity	chart	* - bubble	no interact.	SVG	free	http://goo.gl/6fa2Y5	2015
PhiloGL	3D			WebGL	free	http://goo.gl/hX6IDj	2011
PlotKit	chart	line, pie, bar	no interact.	Canvas, SVG	free	http://goo.gl/zP31MF	2006
Polymap	map	chorop.,symnbol	low-mid	SVG	free	http://goo.gl/HFoYYp	2011
processingJS	general			Canvas	free	http://goo.gl/Gsh0hF	2015
pubnub.wordcloud	text			SVG	free	https://goo.gl/FIFwk3	2015
raphael	general			SVG	free	http://goo.gl/u91nbB	2015
RGraph	chart	bar, line pie, radar	no interact.	Canvas	free	http://goo.gl/6swCiw	2015
Rickshaw (d3)	chart	bar, line	low-mid	SVG	free	http://goo.gl/eMRlAK	2015
sDashboard (Flotr2)	chart	bar, line, pie, bubble	low-mid	Canvas	free	http://goo.gl/L11wz3	2015
sheetsee.js (d3)	chart, map	line	low-mid	SVG	free	http://goo.gl/6GUup5	2015
Shield UI	chart			SVG	comm.	http://goo.gl/nU9sEh	2015
Sigma	graph	force	low-mid	Canvas	free	http://sigmajs.org/	2015
Simplify.js (Leaflet)	chart	line	no interact.	SVG	free	http://goo.gl/Kq0TN0	2015
smoothie.js	chart	line	no interact.	Canvas	free	http://goo.gl/yF0bN6	2015
Springy.js	graph	force	low-mid	Canvas, SVG	free	http://getspringy.com/	2015
TeeChart	chart	line	no interact.	Canvas	both	http://goo.gl/igPMu3	2013
three.js	3D			Canvas, SVG, WebGL	free	http://threejs.org/	2015
Timeplot	chart	line	low-mid	SVG	free	http://goo.gl/7B2Mm4	2009
Toxiclibs.js (raphael, three.js, processingJS)	comput.plot.			Canvas, SVG, WebGL	free	http://goo.gl/KRDf8N	2015
TufteGraph	chart	bar	no interact.	SVG	free	http://goo.gl/asw5P2	2013
vega (d3)	chart, map, graph, text	no interact.		Canvas, SVG	free	http://goo.gl/LiS7jg	2015
vis.js	chart	bar	no brush	Canvas, SVG	free	http://visjs.org/	2015
VivaGraphJS	graph	force	no interact.	SVG	free	http://goo.gl/sH1yvc	2015
xchart (d3)	chart	bar, line	low-mid	SVG	free	http://goo.gl/qPSQaF	2013
XKCD plots (d3)	chart	line	no interact.	SVG	free	http://goo.gl/jj7cAs	2012
YUI chart	chart	line	no interact.	Canvas, SVG	free	http://goo.gl/z8jhy7	2015
ZingChart	chart			Canvas, SVG	comm.	http://goo.gl/yrC49u	2015
Zoomchart	chart			Canvas	comm.	http://goo.gl/uHuxzx	2015

Chapter 10

Redesigning FreeBu and D-Explorer

Previously we have in part addressed issues on online social privacy and aggregate data transparency by developing interactive/exploratory visualization tools, namely FreeBu and D-Explorer. Informed by the previous experience from user studies and the investigation of data-visualization libraries, in this chapter, we further identify the limitations of the two tools, and improve them by redesigning the visualizations, user interactions and the corresponding libraries.

10.1 Redesigning FreeBu

To accommodate users' friend-grouping strategies, FreeBu#3 consists of four views (Section 6.3). The circle view, with the modularity-based community detection algorithm, supports the "inner-circle" strategy, which refers to that users think of the mutual connections between friends when they try to categorize friends. The map view supports both the "inner-circle" strategy and the "mutual-friend" strategy. It directly visualizes the user's ego-network with a force-directed layout, so that the user can inspect the clustered nodes with mutual connections and the individual nodes that connects/bridges other parts of the network. The column view supports the "shared-community" strategy with the user's friends grouped into "columns" of different attributes. The rank view in part supports the "contact-type" strategy by approximating relationship closeness with ranked chat frequencies.

Next, we identify and summarize the improvement points for FreeBu#3 in Subsection 10.1.1, and describe the redesigned and reimplemented FreeBu in detail in Subsection 10.1.2 and 10.1.3.

10.1.1 Points for Improvement

Technology Choice

In the subsequent user study (Section 6.4), we found that the overall usability was satisfactory for the users – the average ratings were above 4 on a 7-Likert scale (Appendix B). However, we also found that the average score on the difficulty to use FreeBu#3 was relatively low (4.07 for USAB 2 and 4.17 for USAB 12, see Appendix B), meaning that some users considered the tool difficult to use. As discussed in Section 6.6, this was partially due to that computing and visualizing an user's entire ego-network were resource-intensive when the network has thousands of nodes. Later we also found that library that we used at the time, ProcessingJS, was incompatible with certain browsers, making users confused when a visualization does not work properly. Furthermore, ProcessingJS does not support object-oriented user interaction, the developer needs to manually program mouse-event detection within the boundary of a visual object, as elaborated in Section 9.4. We adopted another technology, SVG-based graphics, instead of the Canvas-based graphics to take the advantage of SVG's object-oriented event listening. Also, since each SVG element is part of the native HTML DOM tree, debugging becomes more straightforward within browsers. We used the popular SVG-based D3 library to append visual objects, calculate positions and other visual encodings.

Attribute Selection

The user study also revealed that the column view was not as well received as the other views. This was partially due to that the selected attributes were not considered very relevant, including age, school, work, location, language (as shown by the relatively low scores on the item COL 4 and COL 5 in Appendix B). Indeed, for the "shared-community" strategy, important attributes such as neighbour, family, interest group, social group (e.g. youth organization) were missing in our Facebook data, and therefore were not present in FreeBu#3. Another reason that the column view was less favored, we extrapolate, is because of its form of visualization. Though considered relatively well arranged (COL 1), the columns do have the problem of spanning too long, the user has to scroll to the far right to see the end of the stacked columns, of which the attributes

are not very relevant in the first place. The core issue is that the view lacks emphasis, or organization of attributes, the user needs to go through a large number of attributes to find the ones that they consider relevant. Therefore, we decide to remove the column view, and instead create attribute check boxes on the side the other visualizations. The user can check the boxes to filter friends based on attributes. This is inline with the "Filter & Search" improvement point. Furthermore, we limit out attribute selections to "gender", "hometown", "location" and "education", as these categories of attributes were interacted relatively more than the other attributes (Section 6.5).

Filter & Search

We consider seven categories of user tasks [154]: overview, zoom, filter, details-on-demand, relate, history and extract. All the four views provide overviews of the visualized data. The circle view and the map view provide continuous zooming, the rank view provides a two-level discrete zooming, while the column view, due to the nature of its visualization, does not have zooming. All friend nodes in the four views have "tips", which can be brought up by mouse-over, though the content of the tips is merely friend names. A little more information could be helpful for reminding the user of the identify of a friend. However, too much information should be avoided for it can easily clutter the visualizations. Moreover, Facebook already offers a sophisticated platform for its users to look into the details of a friend. The four visualizations also present different perspectives on the structure of a user's ego-network in terms of grouping criteria, which is one approach to provide users the opportunity to relate information items in his ego-network. The user can also compose customized friend lists can submit them to his Facebook account (as the "extract" function). We do not implement "history" functions, because FreeBu# is a relatively light-weight exploratory visualization tool, customized friend lists can be straightforwardly named, created and removed. The usability question items (Appendix B) also showed that users can easily recover from errors (USAB 4 and USAB 12).

However, for the filter-related functions, FreeBu#3 indeed requires upgrade. As addressed in Section 6.6, search is yet to be implemented, for the user may need to search for a particular piece of information. The search function falls under a bigger category of user tasks when interacting with information visualizations: filter. Other filters based on different attribute values should also be considered. This has been addressed in the previous point.

Layout Arrangement

As addressed in Section 6.6, in the circle visualization, we notice that sometimes the user needs to zoom-in fairly deeply to reveal the friend names in a circle. Both the label-revelation threshold and the positioning of the name labels need adjustment. Also, a bigger issue is the relatively ineffective use of space, which results in frequent zooming in and out, and panning. We can counter this issue by adopting a more "space-filling" visualization that also preserves the "hierarchical structure" of the grouping data.

Also, as discussed in Section 6.6, the graph layout in the map visualization was considered less spatially orderly compared to the other visualizations (Appendix B). We can alleviate this problem by introducing color coding and "collapse/expand" features that organize the nodes in the same community into larger, singular nodes. More specifically, when the network is "expanded", all the nodes and links are shown, when the network is "collapsed", only the "backbone" nodes and links are shown, and the degree of this collapsing should be controlled by the user.

Visualization Consistency

Consistency is important for multi-view visualizations [179] and interface design in general [153]. Consistency refers to "common action sequences, terms, units, layouts, colors, typography, and so on within an application program" [153]. We have used the same typography and the same "drag-drop" list-composing interface to promote consistency across the visualizations in FreeBu#3. But we did not take other consistency factors into account, such as color, shape and transitions. For example, a friend node is represented as a grey circle in the circle view, a light blue circle in the map view, a randomly colored rectangle in the column view, etc. When the user switches from one view to another, old visual objects are removed immediately, then new ones are drawn, there lacks transitions that inform the user that the same set of friend nodes are being drawn, only in different representations. Animation should be implemented to show such transitions.

API Independency & Generalizability

Up till FreeBu#3, the tool has been dependent on Facebook graph API. While this provides the convenience for users to directly log into the tool with their Facebook accounts and export customized friend lists to their accounts, the tool itself becomes difficult to be maintained as the API changes. Recently there has

been major API change[1] that limits the retrieval of a Facebook user's friends' data by a third-party application. It, along with other minor changes, made FreeBu#3 dysfunctional.

We realize that it is important for FreeBu to be independent of Facebook graph API, or any other API that is subject to continuous, major change and beyond our control. Also, FreeBu can be generalized to visualize any graph data with/without attribute values specified on nodes and links. A generic exploratory visualization tool is indeed more valuable than a mere Facebook plug-in.

10.1.2 Data Specification

Graph Data

In order to make FreeBu generic, we define a simple format for the input data. As an online tool, FreeBu is written mainly in Javascript, and JSON is the standard language that Javascript uses to communicate data, we continue to use JSON as our input data format, as other Javascript-based APIs do. An example dataset is shown in Listing 1, it has two array: "nodes" and "links". The "nodes" array contains a list of nodes, in the case of Facebook ego-network, these are friend nodes. The "links" array contains the list links among the nodes. A link must have "source" and "target" items that contain the indices that refer to the node lists, in the order of the appearances of the nodes (e.g. line 55-56). A link can further contain customized items such as "weight" (line 57).

Each node must have three keys: "id", "name" and "attributes". The "id" value needs to be unique, the "name" is used to label the node in the visualizations, and the "attributes" is a list of attributes that are used in filter and search. An attribute must have the following keys: "id", "name", "type" and "value". An attribute can be of the types: "info", "polynary", "binary" and "numeric". The "info"-typed (e.g. line 10) attributes are used in the tip that appears on top of a node during mouse-hover. The type "polynary" refers to an attribute with more than two categorical values in all the nodes (e.g. line 16). Similarly, the type "binary" refers to an attribute with with two categorical values (e.g. line 34). The type "numeric" refers to an attribute with numerical values (e.g. line 22 and 28). An attribute can be both "info" and any of the types "polynary", "binary" and "numeric". The "value" item of a "binary" attribute is an array, which may contain multiple values (e.g. line 35-41). Note that we pre-computed the

[1] `https://developers.facebook.com/docs/apps/changelog`

communities, betweenness values and chat frequencies in our example dataset, and we will be using an ego-network dataset for demonstration henceforth.

Listing 1: An example on the graph data specification for FreeBu input.

```
{
  "nodes": [
    {
      "id": "1940985",
      "name": "Sharyn",
      "attributes": [
        {
          "name": "birthday",
          "id": "birthday",
          "type": ["info"],
          "value": "07/31"
        },
        {
          "name": "community",
          "id": "comm",
          "type": ["polynary"],
          "value": "RABB"
        },
        {
          "name": "betweenness",
          "id": "between",
          "type": ["numeric"],
          "value": 0.051818691194057465
        },
        {
          "name": "chat freq.",
          "id": "chat_freq",
          "type": ["numeric"],
          "value": 0
        },
        {
          "name": "education",
          "id": "edu",
          "type": ["binary"],
          "value": [
            "NYC Charter High School, U.S.",
            "Groep T College Leuven",
            "University of Illinois at Urbana-Champaign",
```

```
            "KU Leuven",
            "UCLA"
          ]
        }
      ]
    },
    {
     "id": "8637268",
      "name": "Providencia",
      "attributes": [
          ...
      ]
    }
  ],
  "links": [
    {
      "source": 0,
      "target": 1,
      "weight": 3.5
    },
     ...
  ]
}
```

Hierarchical Data

We also need another pre-calculated hierarchical data as input. This is to reduce the computation during user interaction in the hierarchical view of the new FreeBu, as elaborated in the next subsection. This data is also in JSON format, an example is shown in Listing 2. A non-leaf node in the hierarchy has two keys: "name" with a string value and "children" with an array. A leaf node is one of the nodes in the graph input data, only with the "id" and "name" values. In our demonstration, we use the modularity-based community detection algorithm iteratively to derive the hierarchical dataset. It corresponds to the hierarchical circles in the circle view of FreeBu#3.

Listing 2: An example on the hierarchical data specification for FreeBu input.

```
{
  "children": [
    {
      "children": [
```

```
        {
          "name": "Antonina",
          "id": "818109931"
        },
        {
          "name": "Maye",
          "id": "1280895426"
        }
      ],
      "name": "comm_1"
    }
  ],
  "name": "root"
}
```


10.1.3 Visualization and Interaction

In this subsection, we describe the details of the design of the new FreeBu[2].

FreeBu has three views: the network view, the hierarchical view and the grid view. Figure 10.1 shows the default view — network view. These views correspond to the map view, the circle view and the rank view in FreeBu#3. The interface of FreeBu has three parts, (1) the control panel on the top that consists a line of control buttons, (2) the filter panel on the right that consists of the search box, list-creation buttons and filtering checkboxes, and (3) the main visualization window int he middle.

The checkboxes in the filter panel are constructed based on the node attributes in the input data. For instance, in our example dataset, the only polynary attribute is "community", the corresponding checkbox is automatically created below the "Export Lists" button, followed by the checkboxes on the numeric attributes "betweenness" and "chat frequency", then the binary attributes. The user can zoom and pan the visualization in each view.

Network View

By clicking the "Network View" button, the user is directed to corresponding visualization and provided with corresponding buttons in the control panel. The graph nodes and links are positioned by a force-directed layout. Mouse-hover

[2]The source code is available online, `https://github.com/beaugogh/freebu`. We refer to it simply as FreeBu henceforth.

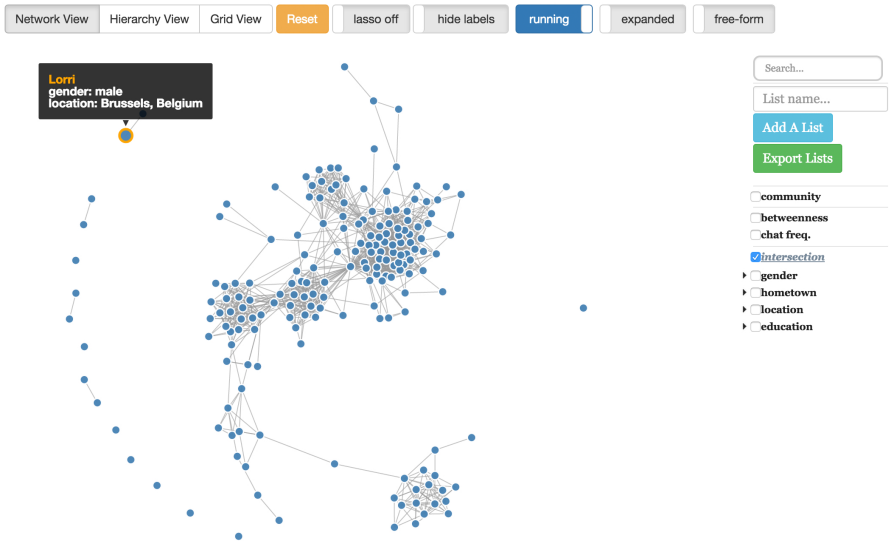

Figure 10.1: The Network View in FreeBu

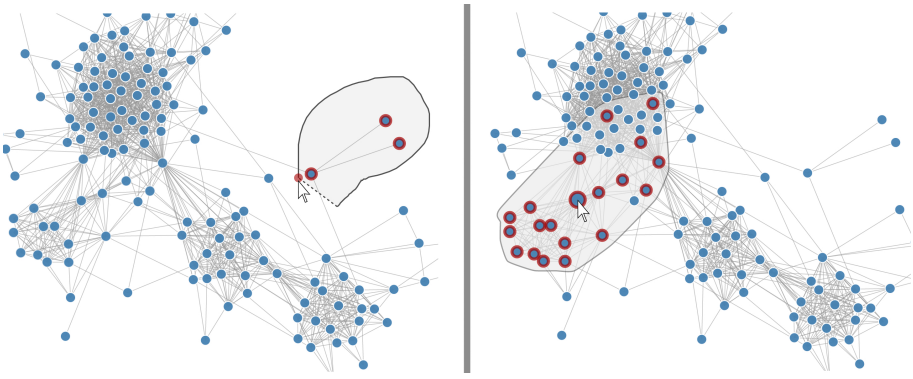

Figure 10.2: Lassoing in the Network View of FreeBu

a node brings out the tip showing more detailed information about that node. The user can draw an enclosing around or on the nodes to make selection in the "lasso" mode, as shown in Figure 10.2 (left). In this mode, the user can also double click a node to select both the node and its neighbours, as shown in Figure 10.2 (right). The "free-form" toggle button on the far right of the control panel is used to indicate if the node positions in the force-directed layout should be contained within the rectangle area in the main visualization.

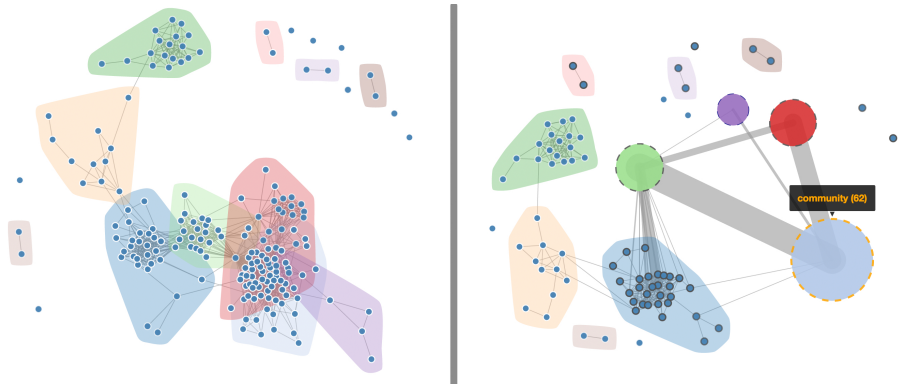

Figure 10.3: Collapsing Mode in the Network View of FreeBu

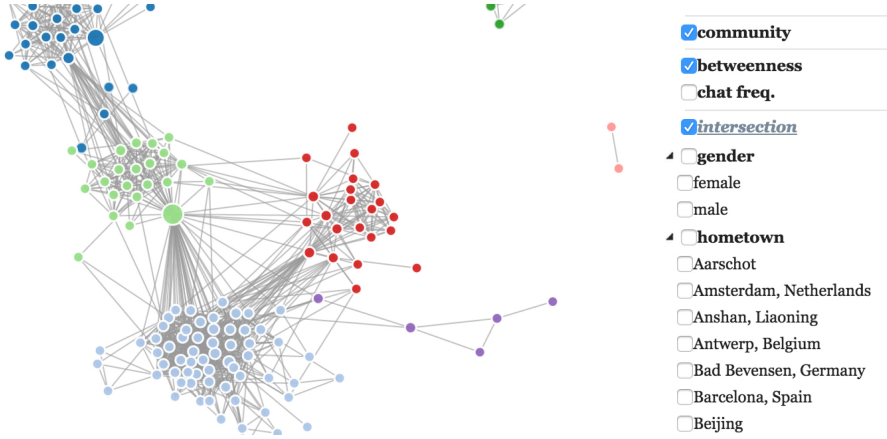

Figure 10.4: Color and size coding in FreeBu

To make the graph layout more organized and manipulatable, we introduce another mode: collapsed mode. The user can toggle the correspondingly named button in the control panel to switch between expanded and collapsed modes. Figure 10.3 (left) shows the collapsed mode where different regions of the graph are enclosed and color-coded with polygons. These regions or clusters of nodes are identified by the "community" attribute in the nodes. This also means that when constructing one's own graph dataset, the "community" attribute needs to be there for the collapsed mode to work. The user can double-click on one of the polygons to collapse the corresponding cluster of nodes into a single "super node", as shown in Figure 10.3 (right), where the sizes of the super nodes and links are accordingly scaled. This can help the user reduce the complexity of the

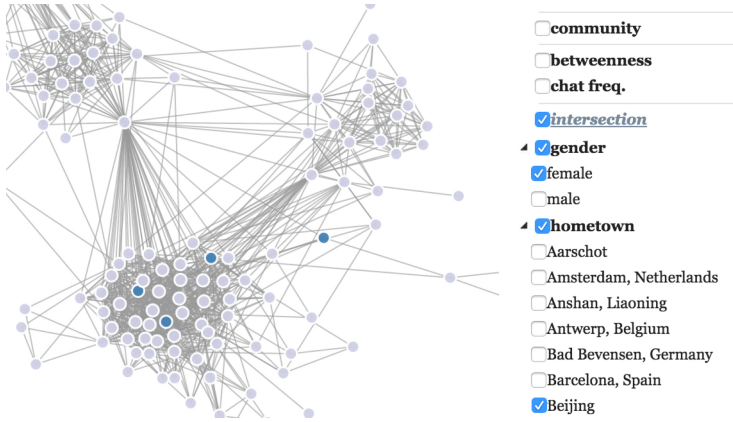

Figure 10.5: Filtering in FreeBu

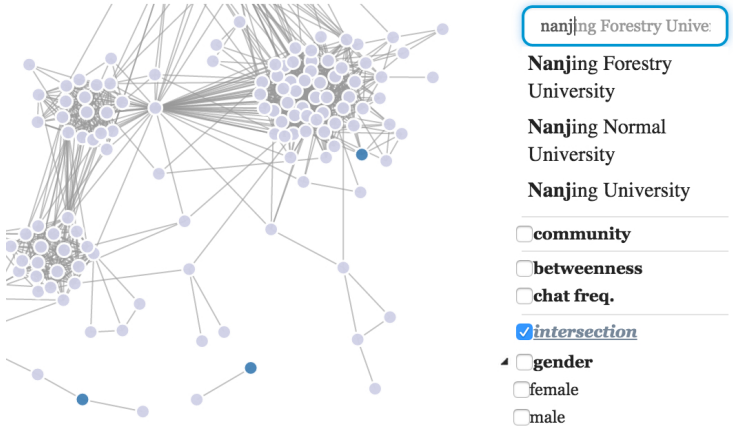

Figure 10.6: Searching in FreeBu

visualization, and reveal the backbone structure of the network. Double-click a super node expands it back. Mouse-hover a super node brings out the tip that shows the number of nodes within it.

The visualization also responds to the changes in the checkboxes in the filter panel. Figure 10.4 shows the nodes are categorically color coded according to the community attribute, and the sizes of these nodes are scaled proportionally to their betweenness scores. Figure 10.5 shows the nodes are filtered based on the selected binary attribute values, namely “gender: female” and “hometown: Beijing”. When checkbox “Intersection” is checked, the nodes that satisfy these

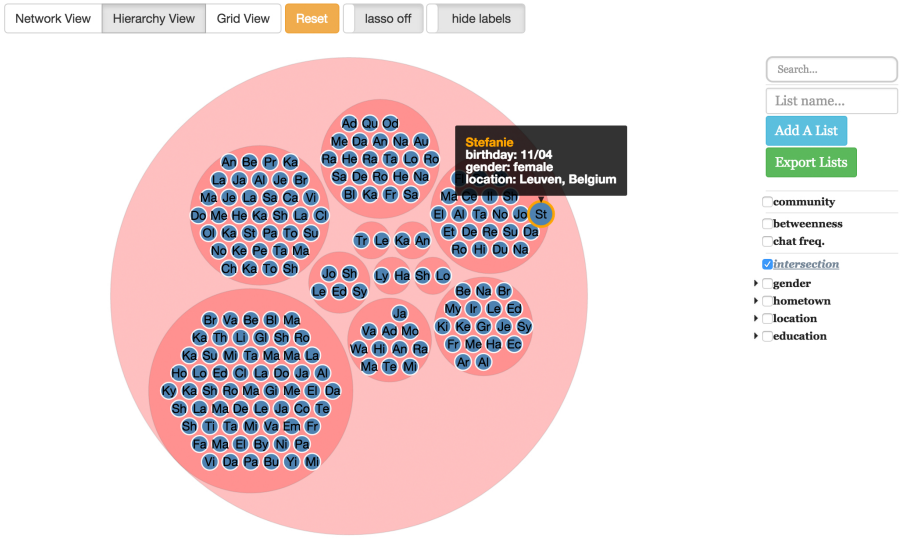

Figure 10.7: The Hierarchical View in FreeBu

two conditions at the same time will be highlighted with deep blue color, other nodes nodes' color is reduced to light greyish blue. If the "Intersection" box is not checked, the nodes that satisfy any of the checked conditions will be highlighted. Figure 10.6 shows the searching interaction in the network view of FreeBu. When the user types in characters in the search box, FreeBu conducts a search through the nodes' names and attributes, finds matchings and returns them as recommended items beneath the search box. Similar to the binary attribute filtering, the matched nodes appear highlighted. The interactions with the checkboxes and the search box apply to all the three views.

Hierarchical View

By clicking the "Hierarchical View" button, the user is directed to this view, as shown in Figure 10.7. This view uses the hierarchical input data to calculate the positions of the nodes. Each node is appended with its text label, of which the number of characters is reduced according to the size of that node. At first, only the top level of communities are displayed. The user can choose to divide a community by clicking on the non-leaf red circles, as shown in Figure 10.8. Clicking a divided non-leaf circle brings it back to its merged state.

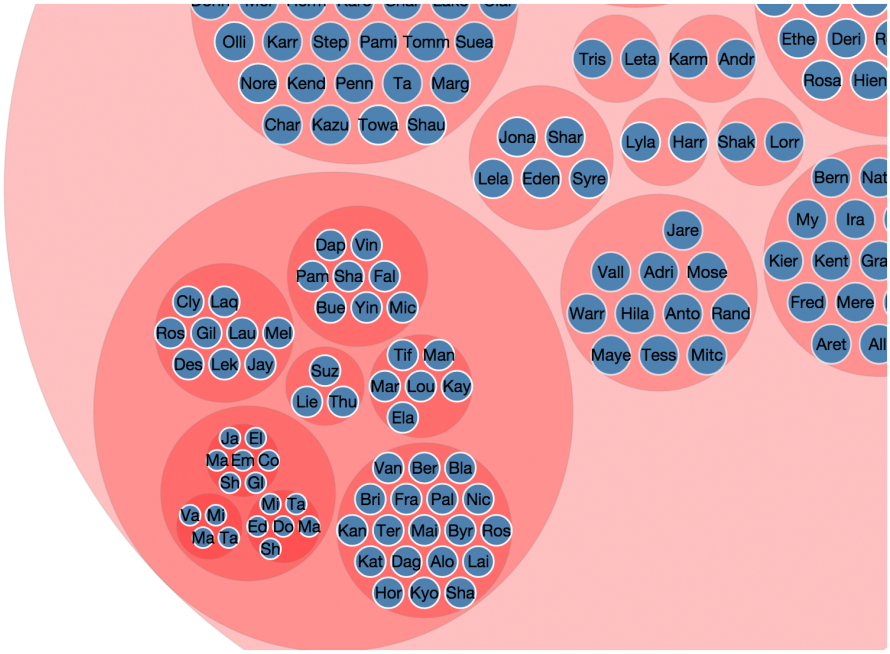

Figure 10.8: Circle Division in the Hierarchical View of FreeBu

Grid View

By clicking the "Grid View" button, the user is directed to this view, as shown in Figure 10.9. This view is especially useful for the user gain insight into the distribution of certain attribute values. The user can sort the nodes according to the selected attribute in the filter panel. The sorting applies to both categorical and numerical values. The nodes in Figure 10.9 are sorted according to the attribute "chat frequency". The nodes in Figure 10.10 are sorted according to the attribute "gender", the nodes with "male" value are highlighted.

List Construction

The three views also share the same list-construction process, as illustrated in Figure 10.11. Similar to that in FreeBu#3, the user creates a new list by clicking the "Add A List" button and can drag and drop individual nodes or grouped nodes into the list. The nodes can be grouped by "lasso" in both the network and grid views, "polygon in collapsed mode" in the network view and

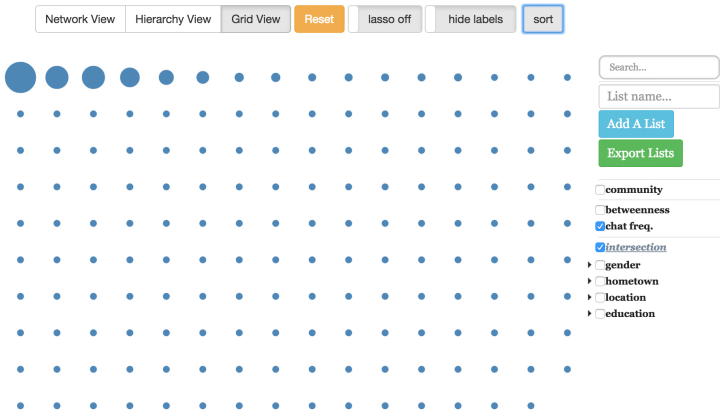

Figure 10.9: The Grid View of FreeBu, with the nodes sorted according to "chat frequency"

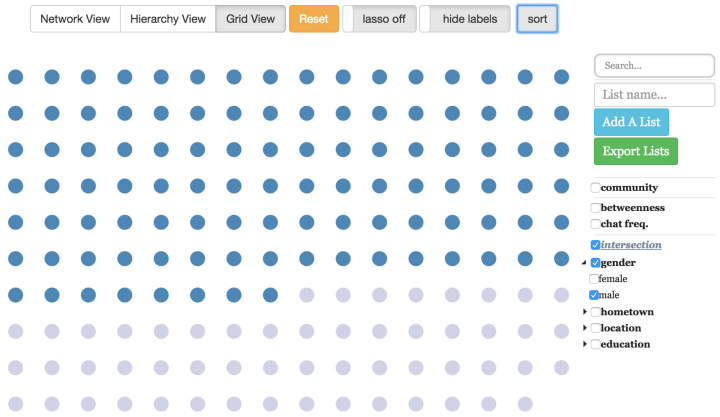

Figure 10.10: The Grid View of FreeBu, with the nodes sorted according to "gender"

"non-leaf circle" in the hierarchical view. The customized lists can be exported to an one-level hierarchical data in JSON by clicking the "Export Lists" button.

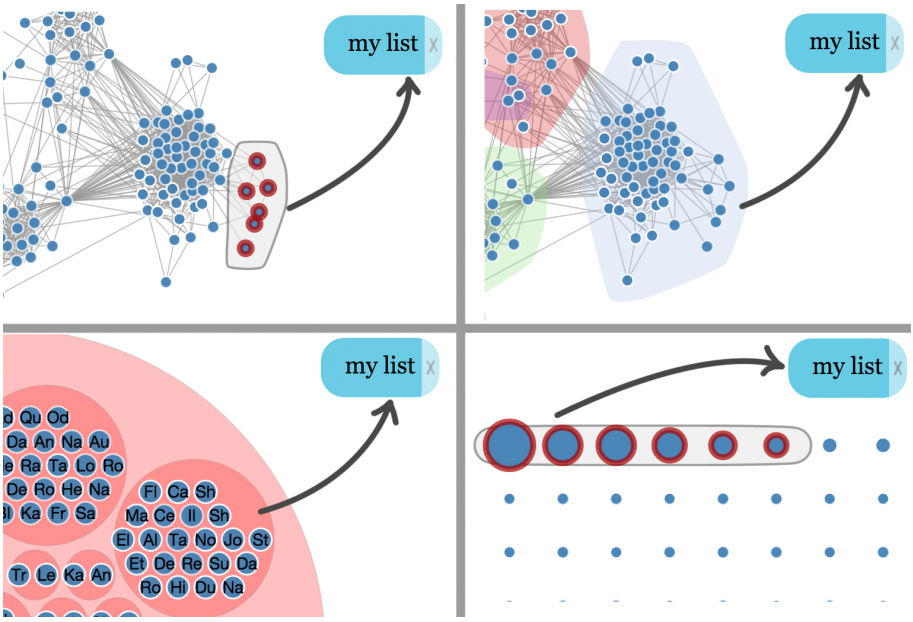

Figure 10.11: Creating lists in FreeBu

10.2 Redesigning D-Explorer

In Chapter 8, we detailed the development of D-Explorer. Operating on the output of the discrimination-aware rule mining system DCUBE, it summarizes and visualizes higher-level information of potentially discriminatory rules. It can be readily extended to classification rule mining method in general. D-Explorer shows the distribution and linkage of items and rules in terms of measures such as support, confidence and lift. In the following, we describe the revised version of the tool. In this revision, similar to FreeBu, D-Explorer is made more generalizable and its visualizations are improved.[3]

10.2.1 Data Specification

We make D-Explorer more generalizable by specifying a simple data format, in which any data can be visualized by the tool. The format follows the CSV (Comma Separated Values) convention. Listing 3 shows an example. The first

[3]The source code of the revised D-Explorer is available online: `https://github.com/beaugogh/d-explorer`.

row is the header, with the names “aset”, “bset”, “class”, “conf” and “meas”, “aset” and “bset” columns contain the attribute-value items in the classification rules. Within one set, the items are separated by space ‘ ’. The reason we differentiate between A set (“aset”) and B set (“bset”) is to accommodate the output of DCUBE, where A set is the items marked by the user as Potentially Discriminatory (PD) items, the remaining items as B set — Potentially Non Discriminatory (PND) items. When there is no need to differentiate two, either the “aset” column or “bset” can be left empty. Each data row (e.g. line 2-5 in Listing 3) represents a classification rule. The “conf” field stores the confidence of the rule, the “meas” field stores any other measure of the rule, e.g. *lift*, or *slift* in DCUBE.

Listing 3: An example on the CSV data specification for D-Explorer input.

```
1 aset,bset,class,conf,meas
2 A1=A1_1,B1=B1_1 B2=B2_1,CLASS=C1,0.53,7.01
3 A1=A1_2,B1=B1_2 B3=B3_1,CLASS=C1,0.66,6.23
4 ,B2=B2_1 B3=B3_1,CLASS=C2,0.35,6.17
5 A2=A2_3 A3=A3_1,,CLASS=C3,0.20,1.46
```

10.2.2 Visualization and Interaction

In the initial D-Explorer, we used a rainbow color scale to color the bubbles and the treemap, as we assumed that people are familiar with the color progression in the rainbow spectrum. However, this may not be the case. Furthermore, as summarised in [159], yellowish colors have a highlight effect that interferes with showing extreme values, and red and violet colors can also be perceptually similar. Therefore, we adopt a color scale with single hue change from “color brewer” [89], which is a tool designed to provide color scales that are perceptually distinct for humans.

Figure 10.12 shows the bubble view in the revised D-Explorer, based on the same dataset RS1 (Rule Set 1) in Section 8.3. Each bubble represents a rule item. The user can choose to “re-color” or “re-layout” the bubbles according to the item measures: support, confidence (average) and measure (average). In the example, the bubbles are scaled and positioned according to “support”, and their colors are scaled according to “measure”. Figure 10.13 shows the bar view with the same setting. The PD items are highlighted with red strokes. We can see that the effect is similar to that of Figure 8.2, but the color contrast is more prominent, in the sense that the user can immediately see that though the PD items $FOREIGN\_WORKER = yes$ and $PERSONAL\_STATUS = female\_div\_sep\_mar$ (divorced, separated or married female) are frequent,

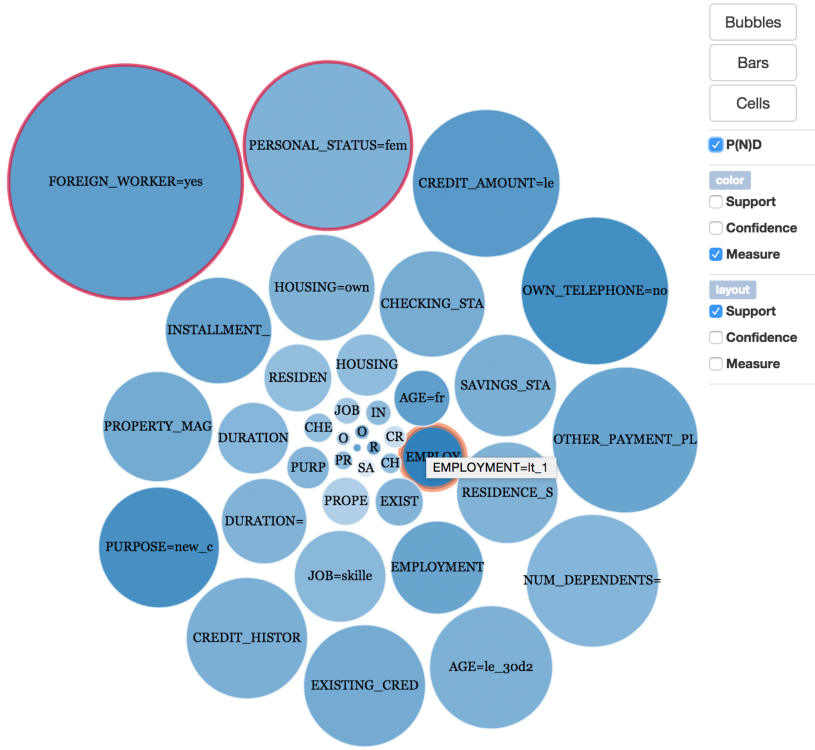

Figure 10.12: The Bubble View in D-Explorer

the items such as $EMPLOYMENT = lt\_1$ (employment is less than one year), and $PURPOSE = new\_car$ (the purpose of the loan is to buy a new car) are more "concentrated" in strong discriminatory rules, in terms of their average discrimination scores.

For showing the associations among rule items, we chose to align the bubbles, and draw arcs connecting pairs of bubbles. The widths of the arcs are proportional to the corresponding mutual information values. However, aligning bubbles on a single horizontal line is not space-efficient, when there are simply too many items, some bubbles would become too small to see. More importantly, there is often a large number of arcs, with similar widths. This could easily introduce occlusion. We therefore adopt a matrix-heatmap view, in which matrix cells are colored according to the pairwise mutual information values. Figure 10.14 shows an example. The item names are aligned both horizontally and vertically as the legends for the matrix. The cells in the matrix are colored according to the mutual information value between each pair of items. The cells of self-

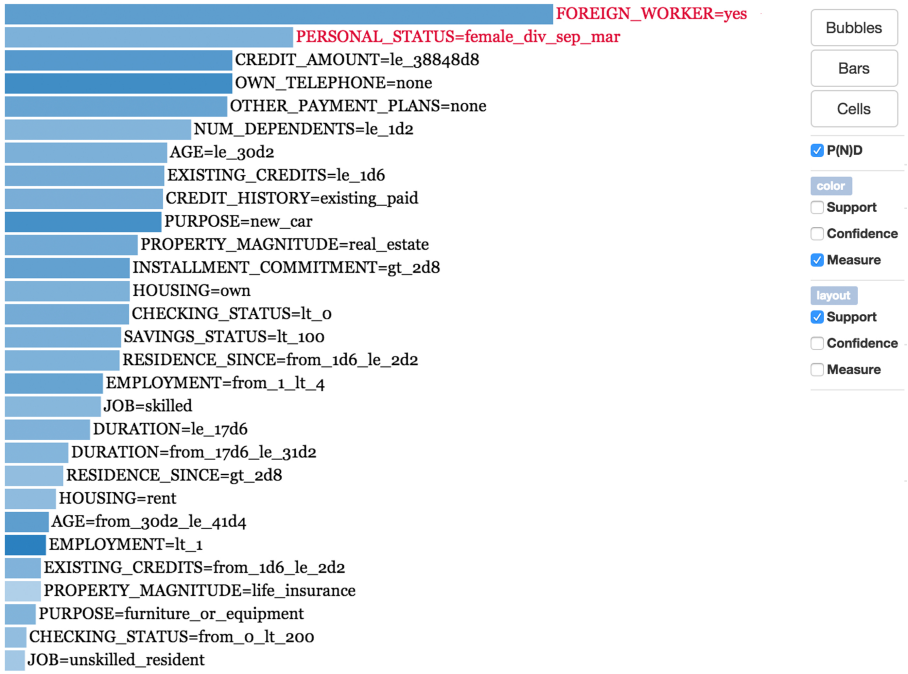

Figure 10.13: The Bar View in D-Explorer

pairs and the pairs with the mutual information value ≤ 0 are colored white. The cells of the pairs with positive mutual information are colored on linear, single hue scale. When the user mouse-hovers a cell, the cell is highlighted, and the information about the corresponding pair is shown on the right. For example, we can see that the items $PROPERTY\_MAGNITUDE = car$ and $OWN\_TELEPHONE = yes$ are strongly related.

The benefit of this type of matrix heatmap is that the user can manipulate the order of the legend items, so that the cells are rearranged and new patterns appear. To this end, we implement two types of re-ordering. When the user checks one of the three boxes in the layout section on the right, the legend items (both horizontal and vertical) are sorted according to the selected criteria. Figure 10.15 gives an example where the legends are sorted based on the "supports" of the items, from which we can observe that the items with larger supports tend to form more pairs with positive mutual information, but there are a few pairs with lessor supports display strong mutual information.

The user can also sort the columns or rows of the matrix by clicking on

Figure 10.14: The Matrix View in D-Explorer

a legend item name. Figure 10.16 shows an example where the columns are sorted according to the mutual information values between the clicked item $FOREIGN\_WORKER = yes$ and the others, in a descending order. From this re-arrangement, we can immediately see that this item is associated with most of the other items, but its association with the other PD item $PERSONAL\_STATUS = female\_div\_sep\_mar$ is not as strong as its associations with most PND items. Moreover, the strongest association is found with the item $OTHER\_PAYMENT\_PLANS = bank$, which also associates with few others, such as $PURPOSE = new\_car$, if we look vertically.

10.3 Conclusion

Informed by previous studies, we identified improvement points for the two exploratory data visualization tools: FreeBu and D-Explorer. We then

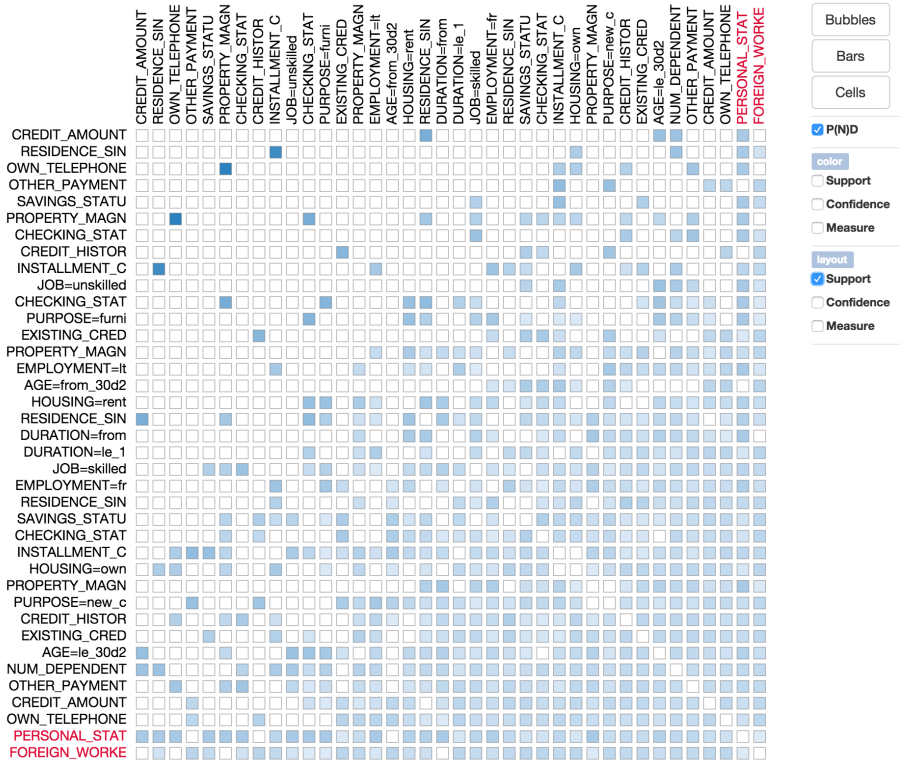

Figure 10.15: Matrix cells arranged according to “support” in D-Explorer

proposed new designs and described the development of these two tools. Users can visualize and explore their own datasets created according to our data specifications, or utilize the library codes downloadable on the websites.

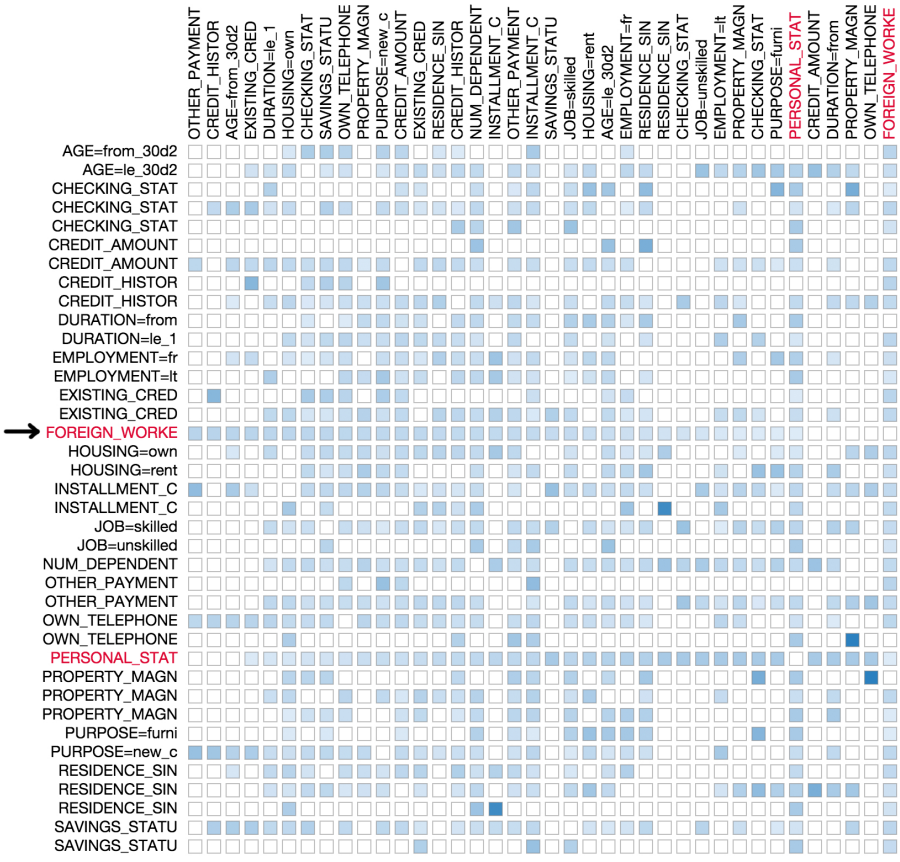

Figure 10.16: Matrix cells arranged according to “mutual information” in D-Explorer

Chapter 11

Conclusion

11.1 Summary

In this thesis, we start from the concept of information literacy, and the need for its curriculum to be extended with privacy literacy and critical data literacy. This leads to a further investigation on the non-conventional, yet necessary notions of privacy and knowledge discovery. In this investigation, we see that privacy is not only about hiding information, but about control and practice. A user gains knowledge about others in relation to himself and re-negotiate the boundaries between his own privacy and the public space.

Feedback and Awareness (FA) tools are online applications for data summary, analysis and exploration towards non-expert users. These tools may provide privacy-related information to users, and promote aggregate data transparency in general. Information visualization plays an important part in FA tools. In this thesis, we mainly focus on exploratory visualization solutions towards online social privacy and aggregate data transparency.

In Chapter 2 we reviewed related work on solutions towards online privacy concerns. We then reviewed privacy-related FA tools in general in different categories. We found that though the FA tools that focus on online social privacy were abundant, few followed the privacy-as-practice approach that allowed the user to gain insight into the data of his friends in the ego-network. And there especially lacked exploratory visualization tools for Online Social Network (OSN) users.

In Chapter 3, we took a first step to address the online social privacy

problem, context collision, by developing a friend-grouping tool (FreeBu#1) with interactive visualization for Facebook users. We first investigated the way that people group the people they know in a user study. And we found that the participants performed the groupings based on relationship type and strength, followed by education, family, workplace, etc. Informed by this result, we then motivated and adopted the Modularity-based community detection method [134] and designed the user interface for the user to have an overview of his ego-network friends and construct Facebook friend lists. The subsequent user study revealed that users considered FreeBu#1 to be valuable in terms of friend-removal, friend-grouping and friend-overview/reflection.

In Chapter 4, we investigated the types of regretted posts from Facebook users in a user study. We found that the disclosure of sensitive photos and emotional or nasty textual posts are often the causes of regrets. We also identified the points of improvement for FreeBu#1 and redesigned it into FreeBu#2. With the interface of this tool, we studied which way of friend-grouping is more useful for Facebook users to avoid regrets: the hierarchical modularity-based community detection in an interactive setting (HMOD), or the existing Facebook smart lists (FSL). We found that the HMOD friend-groups are more useful than FSL in terms of entropy.

In Chapter 5, we compared two state-of-the-art community discovery models: the modularity-based model (MOD) and the Generative Model for Friendships (GMF). We ran the two algorithms on three datasets of ego-networks from Facebook, Google+ and Twitter. The generated communities were compared with the corresponding ground-truth circles that were manually created by the users. We found that MOD matched the ground-truth better than GMF in terms of Reversed Balanced Error Rate (RBER). We further extended MOD to allow overlapping communities to be generated by introducing a threshold called External Belongingness. The extension outperformed MOD. Through both the comparison and the extension, we found that the MOD approach, which was purely graph-based, could approximate users' manually created circles very well. And leveraging the link information in an ego-network graph to produce overlapping communities also proved to be effective.

In Chapter 6, we described FreeBu#3, which was extended from FreeBu#2. This extension was informed by the related user study [47] that looked into Facebook users' friend-grouping strategies. The result was similar to our user study (as detailed in Chapter 3), but with additional insights. We extended the original circle view with three more views: map, column and rank, to accommodate the strategies. The subsequent user study showed that FreeBu#3 was considered fairly usable (average score above 4 on a 7-Likert scale). Friend-grouping and reflection were considered by the participants more valuable than friend-removing. Compared to the other views, the column view was

less favoured by the participants, for that the attributes in columns were not considered relevant. Analyzing the user interaction data with the tool also revealed patterns. For example, the visited percentage of a circle is similar regardless of its size. Users were more attracted to "outliers" in color, size or position. Such discoveries are informative for future exploratory visualization design.

In Chapter 8, we turned to the solution approaches towards data transparency in aggregated data. We developed an explorative visualization tool that assisted the user to discover patterns of mined classification rules on a higher level. The rules were the output of a discrimination-aware data mining system named DCUBE [143]. We proposed measures to capture the meta-level distribution of the rules' scores on support, confidence and other measures.

In Chapter 7, we turned to the field of Online Social Networks (OSNs) again, but took a macro perspective. We investigated how Facebook users with different characteristics expressed sentiments, and how the texts posted on Facebook with different privacy settings exhibited different sentiments. On the one hand, we drew the link between our hypothesis testing and existing sociological research. On the other hand, we proposed and implemented algorithms that extract subgroup comparisons for the purpose of exploratory data analysis. Unlike traditional subgroup discovery that looks for interesting subgroups distinct from the entire population, these algorithms aim to find comparisons between subgroups locally. We consider this work is of value to researchers, exploratory visualization developers, OSN users and organizations who wish to gain insights on sentiment patterns in aggregated personal data.

In Chapter 9, because we realized the importance of online data visualization libraries for developing exploratory visualization tools, and there had been no study on the evaluation of such libraries. We investigated more than a hundred libraries, and developed a comprehensive approach to compare and evaluate Javascript libraries for interactive data visualization. Via this approach, visualization developers can strategically choose the library they need. Visualization library developers can improve or extend existing libraries, or design the next generation of libraries with more informed choices and a more systematic methodology.

In Chapter 10, we described the redesign and reimplementation of the two exploratory visualization tools, namely FreeBu and D-Explorer. This developement was informed by our previous user studies and investigation of the visualization libraries.

In sum, this thesis took an interdisciplinary approach towards OSN social privacy and aggregate data transparency. We iteratively designed and implemented two

exploratory visualization tools: FreeBu and D-Explorer. During this process, we leveraged the knowledge from the fields of information visualization, software engineering, human computer interaction, community detection, subgroup discovery and social science.

11.2 Outlook

Privacy and data literacy are broad domains, on which solely developing exploratory visualization tools has very limited impact. In order to promote people's online well-being in terms of social, institutional and governmental privacy, and information literacy in general, data about individuals needs to be more accessible to the corresponding individuals. The development of exploratory visualization tools should involve users on a large scale. Relevant user tasks and measures should be designed to evaluate the usefulness of such tools in particular use cases. It is worth researching whether and how exploratory visualization tools change people's behavior, in large-scale, longitudinal studies. This process also needs support from sociology, psychology, law, education and economics.

It is also vital for the entire technological ecosystem to be supportive of interactive visualizations. So far, sophisticated, highly dynamic online visualization is still considered a niche, where it remains difficult to implement full-fledged applications. However, as the amount of available personal data is increasing, analyzing and exploring data interactively will become an essential part of our life. Researchers and practitioners should strive to broaden and standardize the visual language for data analysis and exploration, including user tasks, input/output data formats, visual encodings and user interactions.

Furthermore, platforms, especially web browsers, need to consider exploratory visualization as the "technical default". Based on this, we can then go beyond the current generation of relatively simplistic, small scale visualization tools, and build online data visualization applications as well as libraries that are free, more powerful, and publicly accessible. This will eventually promote the exploration and analysis of open data as a social and cultural norm.

Part IV

Appendices

Appendix A

Appendix: Interviews

A.1 Interview Questions

The set of questions asked in the interview study of FreeBu#1 is as follows. The conductor of the interviews is Dr.Ralf De Wolf [45].

- What do you think the grouping is based on?
- Does this grouping make sense to you?
- What is missing in the grouping structure?
- What would you like to see in the grouping structure?
- What do you think about the interface?
- What did you not expect?
- Would you also group your friends like this?
- Which features do you appreciate?
- Do you find Tool easy to use?
- Is there something you find annoying?

A.2 Interview Coding

The coding process consisted of three phases to structure the qualitative data. This process helped with summarizing and analyzing the way the participants valued FreeBu#1. The conductor of the interviews is Dr.Ralf De Wolf [45].

Phase 1: Open Coding	Phase 2: Labeling the codes
Concept	Appearance
Zooming	Usability
Ugly	Appearance
Period	Management
Place	Small-group
Alone	Small-group
Context	Mistakes
Separately	Small-group
Circles	Usability
Contact	Management
Control	Management
Visualization	Appearance
Scary	Emotions
Enthusiastic	Emotions
Feedback	Functionality
Wrong-group	Mistakes
Mistakes	Mistakes
Using	Functionality
Integrated	Usability
Mixed	Big-group
Big-groups	Big-group
Small-groups	Small-group
Color	Appearance
Correct	Functionality
Labels	Functionality
Lists	Functionality
Mixed	Big-group
Mind-map	Appearance
Names	Appearance
Not-handy	Functionality
Unfamiliar	Functionality
Divided	Functionality
Unnecessary	Functionality
Remarkable	Functionality

Phase 1: Open Coding	Phase 2: Labeling the codes
Overview	Functionality
Posting	Functionality
Privacy	Functionality
Searching	Usability
Relevant	Functionality
Scratch	Functionality
Shift	Functionality
Unchanged	Appearance
Changes	Appearance
Mixed	Big-group
Dragging	Usability
Deleting	Functionality
Mess	Appearance
See	Functionality
Meaningful	Functionality

Phase 3: Organizing the codes and making further categories

Appearance
Color
Ugly
Label
Static

Functionality
Overview
Reflecting
Grouping
Deleting
Announcements
New

Emotions
Surprised
Scary

Small group

Management
Period
Contact
Context

Usability
Dragging
Zooming
Integrated
Searching

Big group

Mistakes

Appendix B

Appendix: Survey Summary

USABility	**M (S.D.)**	**Responses**
USAB 1 - Overall, I am satisfied with the USABility of FreeBu	5.00 (1.40)	42
USAB 2 - FreeBu was difficult to use (-)	4.07 (1.64)	42
USAB 3 - It was easy to learn to use FreeBu	5.45 (1.56)	42
USAB 4 - Whenever I made a mistake using FreeBu, I recovered easily and quickly	4.84 (1.46)	37
USAB 5 - The videos with instructions were clear	6.05 (1.28)	40
USAB 6 - The interface of FreeBu was pleasant	5.20 (1.36)	40
USAB 7 - The videos and instructions were helpful when testing FreeBu	5.87 (1.51)	39
USAB 8 - The interface was easy to use	5.38 (1.39)	40
USAB 9 - FreeBu had all the functions and capabilities I expect it to have	5.10 (1.28)	40
USAB 10 - FreeBu helps with making Facebook lists	5.46 (1.24)	37
USAB 11 - It was difficult to find the information I needed (-)	4.63 (1.41)	40
USAB 12 - Whenever something didn't work, it was easy to solve	4.17 (1.63)	36
USAB 13 - Overall, I am satisfied with FreeBu	5.07 (1.33)	42

Perceived Value	**M (S.D.)**	**Responses**
PV 1 - FreeBu helps me remove unwanted audiences on Facebook	4.50 (1.29)	38
PV 2 - FreeBu helps me group my Facebook friends	5.23 (1.25)	39
PV 3 - FreeBu lets me think about who my Facebook friends are	5.56 (1.27)	39
PV 4 - FreeBu clarifies the relationships with others I am not fully aware of	5.40 (1.31)	38
PV 5 - FreeBu indicates close and distant friends	4.97 (1.48)	39
PV 6 - FreeBu helps me share information with specific groups of friends	4.79 (1.44)	38
PV 7 - FreeBu gives a wrong image of my Facebook friends (-)	5.00 (1.15)	39
PV 8 - FreeBu helps me with making Facebook lists	5.00 (1.39)	38
PV 9 - FreeBu does not provide a good overview of my Facebook (-)	4.05 (1.69)	39
PV 10 - Overall I find FreeBu a useful tool	4.69 (1.44)	39
Circle visualization	**M (S.D.)**	**Responses**
CIRC 1 - The circles are well-arranged	4.94 (1.63)	35
CIRC 2 - The circles provided me with a clear image of who my Facebook friends are	4.86 (1.72)	35
CIRC 3 - The circles were pleasant to see	5.09 (1.62)	35
CIRC 4 - The circles could match with a grouping I would make	4.62 (1.78)	34
CIRC 5 - The Facebook friends who were grouped together also belonged together	4.57 (1.67)	35
Map visualization	**M (S.D.)**	**Responses**
MAP 1 - The map was well-arranged	4.14 (1.70)	35
MAP 2 - The map provided me with a clear image of who my Facebook friends are	4.66 (1.85)	35
MAP 3 - The map were pleasant to see	4.43 (1.80)	35
MAP 4 - The map indicated how groups of friends are connected with each other	5.14 (1.68)	35
MAP 5 - The map indicated which groups are completely segregated	5.14 (1.87)	35
MAP 6 - The map provided me with a clear image of who my Facebook friends are	4.68 (1.82)	34

Column visualization	**M (S.D.)**	**Responses**
COL 1 - The columns are well-arranged	4.81 (1.76)	31
COL 2 - The columns provided me with the characteristics of my Facebook friends	4.45 (1.84)	31
COL 3 - The columns were pleasant to see	4.87 (1.71)	31
COL 4 - The characteristics in the columns were relevant	4.10 (1.73)	30
COL 5 - The relationship between the characteristics in the columns made clear who my Facebook friends are	3.83 (1.70)	30
COL 6 - The columns made clear who my Facebook friends are	3.81 (1.64)	31

Rank visualization	**M (S.D.)**	**Responses**
RANK 1 - The ranking was well-arranged	5.83 (1.29)	30
RANK 2 - The ranking provided me with a clear image of whom I chatted with	4.70 (1.84)	30
RANK 3 - The ranking were pleasant to see	5.60 (1.13)	30
RANK 4 - The ranking provided me with a clear image of who my Facebook friends are	4.30 (1.86)	30
RANK 5 - The ranking provided me with how much contact I have with my Facebook friends	4.30 (1.92)	30

Appendix C

Appendix: FreeBu Versioning

C.1 FreeBu#1

Figure C.1: Illustration of FreeBu#1

Visualization star-tree

Algorithms star-tree layout, MOD, label generation with F1 measure

C.2 FreeBu#2

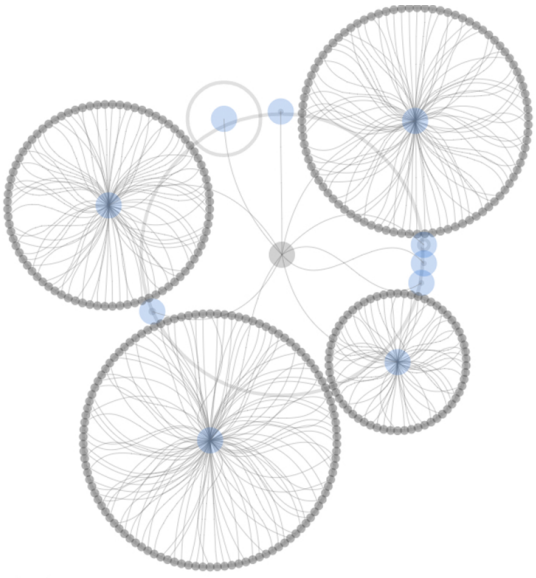

Figure C.2: Illustration of FreeBu#2

Visualization circle-tree

Algorithms circle-tree layout, HMOD

Changes

1. redesigning the star-tree into circle-tree visualization
2. using HMOD instead of MOD
3. removing label generation
4. changing from desktop to online application

Explanations

1. to avoid occlusion in the original star-tree visualization
2. Because sometimes a circle could be overly large, the interactive hierarchical circle visualization provides a way to break down large circles. This also coincides with the human cognitive information processing where conceptual entities are organized as hierarchies.

3. Due to the lack of profile data in an ego-network, the derived labels could be uninformative or misleading, we decided to remove the labelling function in this version of FreeBu, instead, rely solely on friends' names for the user to make sense of a community.
4. to allow the user to directly log in with his/her Facebook account online, and to facilitate automatic input data processing

C.3 FreeBu#3

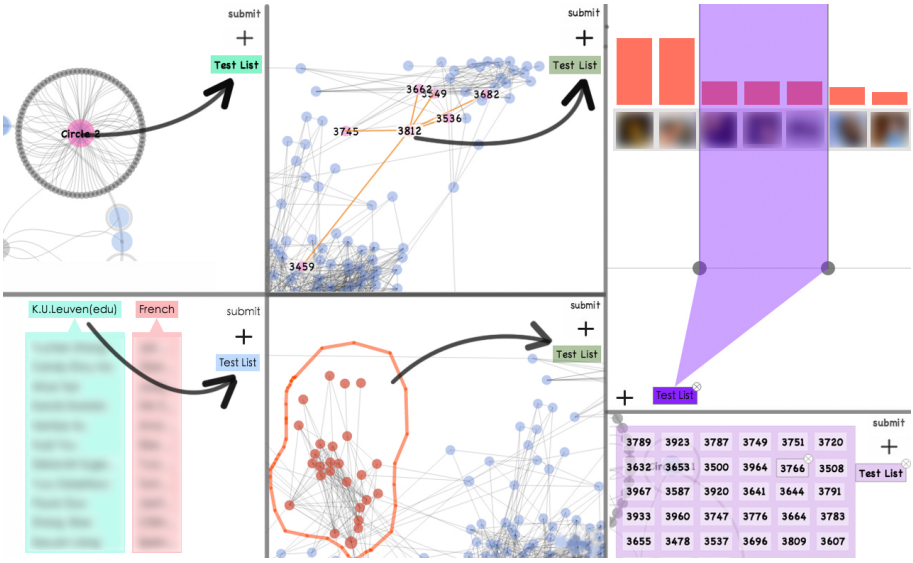

Figure C.3: Illustration of FreeBu#3

Visualizations

A circle view (i.e. circle-tree)

B map view

C column view

D rank view

Algorithms

A circle-tree layout

B HMOD

C force-directed graph layout, Betweenness calculation

D basic categorization according to common attributes

E basic sorting according to chat frequencies

Changes

1. add the map view
2. add the column view
3. add the rank view

Explanations

1. to accommodate the inner-circle and the mutual-friend grouping strategies
2. to accommodate the shared-community grouping strategy
3. to accommodate the contact-type grouping strategy

C.4 FreeBu#4 (or simply FreeBu)

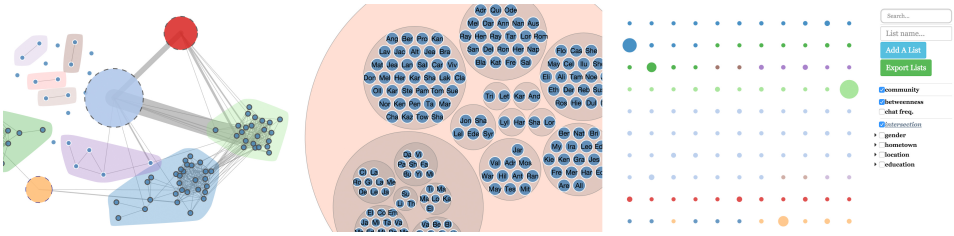

Figure C.4: Illustration of FreeBu#4, or simply FreeBu

Visualizations

A network view

B hierarchy view

C grid view

D view coordination

Algorithms

A force-directed graph layout, community expansion/collapse

B circle-packing layout

C basic sorting according to selected attributes

D attribute ranking, filtering and searching

Changes

1. add community collapsing/expanding utility to the graph layout
2. adopt a new circle-packing layout to replace the previous circle-tree layout
3. modify the previous rank layout with the alignment of nodes in multiple lines
4. remove the column layout, but enable attribute filtering and searching across the other three views
5. define two input formats: json for graph and hierarchy input data

Explanations

1. to enable a better overview and details-on-command utility in the graph layout
2. to make the circles more spacing-filling and easy to interact with
3. to make the aligned visual objects more space-filling
4. to interconnect different views with universal attribute ranking, filtering and searching
5. to standardize the input data preprocessing, making FreeBu a generic graph and hierarchy visualization tool, also to reduce computation during user interaction

Remark: FreeBu is an online application that can be accessed via the URL: `http://freebu.cs.kuleuven.be/`. Upon visiting the website, an example dataset is visualized. For visualizing the user's own dataset, simply drag and drop the file into the "drop area" on the bottom of the page. However, though it is straightforward to develop a crawler that harvests and converts the user's own Facebook data (including the friend graph and friends' profile information) into the dataset with the format specified in Section 10.1, Facebook's Terms of Service[1] forbids this data crawling. We thus cannot provide such a crawler. The user has to construct his/her own dataset, following the data specification (Section 10.1).

[1] `https://www.facebook.com/terms.php`, item 3.2

Appendix D

Appendix: Software Installation

Remark: There exists many softwares that offer free local servers, we choose the AMP (Apache MySQL PHP) stack for illustration.

D.1 FreeBu Installation

To run FreeBu on a local server in your computer, follow the steps below:

1. go to `http://ampps.com/download`, choose the download option for Windows, MacOS, or Linux
2. download and install AMP on your operating system
3. go to the directory where the AMP is installed, find the folder named "htdocs"
4. download the file source code in zip from `https://github.com/beaugogh/freebu`
5. unzip the file downloaded file into the folder "htdocs" and rename it to "freebu"
6. start your local server with the AMP interface
7. go to URL "localhost/freebu" in the browser and you can see FreeBu is up and running

D.2 D-Explorer Installation

To run D-Explorer on a local server in your computer, follow the steps below:

1. go to `http://ampps.com/download`, choose the download option for Windows, MacOS, or Linux
2. download and install AMP on your operating system
3. go to the directory where the AMP is installed, find the folder named "htdocs"
4. download the source code in zip from `https://github.com/beaugogh/d-explorer`
5. unzip the downloaded file into the folder "htdocs" and rename it "dex"
6. start your local server with the AMP interface
7. go to URL "localhost/dex" in the browser and you can see D-Explorer is up and running

Bibliography

[1] Abbasi, A., Hassan, A., and Dhar, M. Benchmarking Twitter sentiment analysis tools. In *The International Conference on Language Resources and Evaluation* (2014), pp. 823–829. pages 112

[2] Acquisti, A., and Gross, R. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Privacy enhancing technologies* (2006), Springer, pp. 36–58. pages 4

[3] Acquisti, A., Van Alsenoy, B., Balsa, E., Berendt, B., Clarke, D., Diaz, C., Gao, B., Gurses, S., Kuczerawy, A., and Pierson, J. Spion Deliverable D2.1 — State of the Art. *SBO Security and Privacy for Online Social Networks* (2011). pages 13, 15

[4] Allport, G. W. *The Nature of Human Prejudice*. Basic books, 1979. pages 35

[5] Asuncion, A., and Newman, D. Uci machine learning repository, 2007. pages 126

[6] Auber, D. Tulip—A huge graph visualization framework. In *Graph Drawing Software*. Springer, 2004, pp. 105–126. pages 36

[7] Backstrom, L., and Kleinberg, J. Romantic partnerships and the dispersion of social ties: A network analysis of relationship status on facebook. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work & social computing* (2014), ACM, pp. 831–841. pages 87

[8] Bacon, K., and Dewan, P. Towards automatic recommendation of friend lists. In *The 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing* (2009), IEEE, pp. 1–5. pages 37

[9] Bakewell, S. *How to live: A life of Montaigne in one question and twenty attempts at an answer*. Random House, 2010. pages 4

[10] Baldwin, T., and Lui, M. Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010), Association for Computational Linguistics, pp. 229–237. pages 109

[11] Balsa, E., Brandimarte, L., Acquisti, A., Diaz, C., and Gürses, S. Spiny CACTOS: OSN users attitudes and perceptions towards cryptographic access control tools. In *USEC* (2014), p. 10. pages 14

[12] Balsa, E., Troncoso, C., and Diaz, C. A metric to evaluate interaction obfuscation in Online Social Networks. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 20*, 06 (2012), 877–892. pages 4

[13] Bastian, M., Heymann, S., Jacomy, M., et al. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the Third International Conference on Weblogs and Social Media (ICWSM)* (2009), pp. 361–362. pages 36

[14] Bayardo Jr, R. J., and Agrawal, R. Mining the most interesting rules. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1999), ACM, pp. 145–154. pages 130

[15] Beard, D., and Walker, J. Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour & Information Technology 9*, 6 (1990), 451–466. pages 58

[16] Becker, R. A., and Cleveland, W. S. Brushing scatterplots. *Technometrics 29*, 2 (1987), 127–142. pages 160

[17] Berendt, B. D5.1-Report on Research Activities. *Communications of the ACM 48*, 4 (2008), 101–106. pages 7

[18] Berendt, B. Data mining for information literacy. In *Data Mining: Foundations and Intelligent Paradigms*. Springer, 2012, pp. 265–297. pages 2, 3, 14

[19] Berendt, B. More than modelling and hiding: towards a comprehensive view of web mining and privacy. *Data Mining and Knowledge Discovery 24*, 3 (2012), 697–737. pages 5, 7

[20] Berendt, B., and Preibusch, S. Exploring discrimination: A user-centric evaluation of discrimination-aware data mining. In *The 12th International Conference on Data Mining Workshops* (2012), IEEE, pp. 344–351. pages 136

[21] Bernstein, M. S., Bakshy, E., Burke, M., and Karrer, B. Quantifying the invisible audience in social networks. In *ACM SIGCHI Conference on Human Factors in Computing Systems* (2013). pages 4

[22] Bhandari, I. Attribute focusing: machine-assisted knowledge discovery applied to software production process control. *Knowledge Acquisition 6*, 3 (1994), 271–294. pages 130

[23] Bisson, G., and Blanch, R. Improving visualization of large hierarchical clustering. In *The 16th International Conference on Information Visualisation* (2012), IEEE, pp. 220–228. pages 57

[24] Blanchard, J., Guillet, F., and Briand, H. Interactive visual exploration of association rules with rule-focusing methodology. *Knowledge and Information Systems 13*, 1 (2007), 43–75. pages 128, 130

[25] Blanchard, J., Pinaud, B., Kuntz, P., and Guillet, F. A 2D–3D visualization support for human-centered rule mining. *Computers & Graphics 31*, 3 (2007), 350–360. pages 128

[26] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment 2008*, 10 (2008), P10008. pages 43, 44

[27] Bostock, M., Ogievetsky, V., and Heer, J. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (2011), 2301–2309. pages 140

[28] Boyd, D. *Faceted id/entity: Managing representation in a digital world*. PhD thesis, Massachusetts Institute of Technology, 2002. pages 49

[29] Boyd, D. M. *Taken out of context: American teen sociality in networked publics*. PhD thesis, University of California-Berkeley, School of Information, 2008. pages 4, 33

[30] Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. On finding graph clusterings with maximum modularity. In *Graph-Theoretic Concepts in Computer Science*, A. Brandstädt, D. Kratsch, and H. Müller, Eds., vol. 4769 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007, ch. 12, pp. 121–132. pages 51

[31] Brehmer, M., and Munzner, T. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2376–2385. pages 30, 150, 158

[32] Bruls, M., Huizing, K., and Van Wijk, J. J. Squarified treemaps. In *Data Visualization 2000*. Springer, 2000, pp. 33–42. pages 36, 57, 135

[33] Calders, T., and Verwer, S. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery 21*, 2 (2010), 277–292. pages 126

[34] Cazabet, R., Leguistin, M., and Amblard, F. Automated Community Detection on Social Networks: Useful? Efficient? Asking the users. In *Proceedings of the 4th International Workshop on Web Intelligence & Communities* (2012), ACM, p. 6. pages 51

[35] Chen, M. C., Anderson, J. R., and Sohn, M. H. What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing. In *CHI'01 extended abstracts on Human factors in computing systems* (2001), ACM, pp. 281–282. pages 92

[36] Chen, Y.-W., and Lin, C.-J. Combining SVMs with various feature selection strategies. In *Feature Extraction*. Springer, 2006, pp. 315–324. pages 76

[37] Cherulnik, P. D. Sex differences in the expression of emotion in a structured social encounter. *Sex Roles 5*, 4 (1979), 413–424. pages 106

[38] Coleman, J. S. Social capital in the creation of human capital. *American journal of sociology* (1988), S95–S120. pages 34

[39] Collins, A. M., and Quillian, M. R. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior 8*, 2 (1969), 240–247. pages 55

[40] Corbin, J., and Strauss, A. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014. pages 49

[41] Cowan, N. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and brain sciences 24*, 1 (2001), 87–114. pages 54

[42] Danezis, G. Inferring privacy policies for social networking services. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence* (2009), ACM, pp. 5–10. pages 33

[43] de Caritat marquis de Condorcet, J. *Sketch for a Historical Picture of the Progress of the Human Mind*. Library of ideas. Weidenfeld and Nicolson, University Microfilms, 1955. pages 2

[44] De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. The mahalanobis distance. *Chemometrics and intelligent laboratory systems 50*, 1 (2000), 1–18. pages 93

[45] De Wolf, R., Gao, B., Berendt, B., and Pierson, J. The promise of audience transparency. exploring users' perceptions and behaviors towards visualizations of networked audiences on facebook. *Telematics and Informatics 32*, 4 (2015), 890–908. pages 11, 46, 205, 206

[46] De Wolf, R., and Pierson, J. Privacy beyond the individual: analysing group based access control models on social network sites from a user perspective. *International Association for Media and Communication Research* (2012). pages 35

[47] De Wolf, R., and Pierson, J. Who's my audience again? Understanding audience management strategies for designing privacy management technologies. *Telematics and Informatics* (2013). pages 81, 84, 200

[48] Dey, A. K. Understanding and using context. *Personal and Ubiquitous Computing 5*, 1 (2001), 4–7. pages 33

[49] Donath, J. Signals in social supernets. *Journal of Computer-Mediated Communication 13*, 1 (2007), 231–251. pages 34

[50] Dunbar, R., and Sutcliffe, A. Social complexity and intelligence. *The Oxford Handbook of Comparative Evolutionary Psychology* (2012), 102. pages 54, 89

[51] Dunbar, R. I. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution 22*, 6 (1992), 469–493. pages 54

[52] Dunn, O. J. Estimation of the medians for dependent variables. *The Annals of Mathematical Statistics* (1959), 192–197. pages 114

[53] Eckersley, P. How unique is your web browser? In *Privacy Enhancing Technologies* (2010), Springer, pp. 1–18. pages 20

[54] Egelman, S., Oates, A., and Krishnamurthi, S. Oops, I did it again: mitigating repeated access control errors on facebook. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (2011), ACM, pp. 2295–2304. pages 101

[55] Fang, L., Kim, H., LeFevre, K., and Tami, A. A privacy recommendation wizard for users of social networking sites. In *Proceedings of the 17th ACM conference on Computer and communications security* (2010), pp. 630–632. pages 18

[56] Farnadi, G., Sitaraman, G., Rohani, M., Kosinski, M., Stillwell, D., Moens, M.-F., Davalos, S., and De Cock, M. How are you doing? Emotions and personality in Facebook. In *Proceedings of the EMPIRE Workshop of the 22nd International Conference on User Modeling, Adaptation and Personalization (UMAP)* (2014). pages 106

[57] Fayyad, U. M. Data mining and knowledge discovery: Making sense out of data. *IEEE Intelligent Systems*, 5 (1996), 20–25. pages 6

[58] Field, D. J., Hayes, A., and Hess, R. F. Contour integration by the human visual system: Evidence for a local "association field". *Vision research 33*, 2 (1993), 173–193. pages 59

[59] Fisher, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine learning 2*, 2 (1987), 139–172. pages 135

[60] Fortunato, S. Community detection in graphs. *Physics Reports 486*, 3 (2010), 75–174. pages 74

[61] Fortunato, S., and Barthélemy, M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences 104*, 1 (2007), 36–41. pages 51, 55

[62] Fruchterman, T. M., and Reingold, E. M. Graph drawing by force-directed placement. *Software: Practice and Experience 21*, 11 (1991), 1129–1164. pages 87

[63] Furnas, G. W. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1986), ACM, pp. 16–23. pages 163

[64] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994. pages 30, 169

[65] Gao, B., and Berendt, B. Visual data mining for higher-level patterns: discrimination-aware data mining and beyond. In *Benelearn 2011. Proceedings of the 20th Belgian Dutch Conference on Machine Learning* (2011), pp. 45–52. pages 12

[66] Gao, B., and Berendt, B. Circles, posts and privacy in egocentric social networks: An exploratory visualization approach. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (2013), ACM, pp. 792–796. pages 11

[67] Gao, B., and Berendt, B. Friends and Circles—A Design Study for Contact Management in Egocentric Online Social Networks. In *Online Social Media Analysis and Visualization*. Springer, 2014, pp. 129–161. pages 11

[68] Gao, B., and Berendt, B. From visualization taxonomies to library designs: A study of javascript libraries for interactive data visualization. *Submitted to IEEE Transactions on Visualization and Computer Graphics* (2015). pages 12, 138

[69] Gao, B., Berendt, B., Clarke, D., de Wolf, R., Peetz, T., Pierson, J., and Sayaf, R. Interactive grouping of friends in OSN: Towards online context management. In *The 12th International Conference on Data Mining Workshops* (2012), IEEE, pp. 555–562. pages 11

[70] Gao, B., Berendt, B., and Vanschoren, J. Who is more positive in private? Analyzing sentiment differences across privacy levels and demographic factors in Facebook chats and posts. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (2015), pp. 605–610. pages 12, 122

[71] Garcia, D., and Thelwall, M. Political alignment and emotional expression in spanish tweets. In *Workshop on Sentiment Analysis at SEPLN* (2013), pp. 151–159. pages 113

[72] Geiselman, R. E., and Bellezza, F. S. Eye movements and overt rehearsal in word recall. *Journal of Experimental Psychology: Human Learning and Memory 3*, 3 (1977), 305. pages 92

[73] Ghoniem, M., Fekete, J.-D., and Castagliola, P. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on* (2004), IEEE, pp. 17–24. pages 57

[74] Goffman, E. *The Presentation of Self in Everyday Life*. Anchor Books, 1745 Broadway, New York, NY 10019 USA, 1959. pages 33

[75] Gonçalves, E. C., Mendes, I. M. B., and Plastino, A. Mining exceptions in databases. In *Advances in Artificial Intelligence*. Springer, 2005, pp. 1076–1081. pages 130

[76] Gosling, S. D., Gaddis, S., Vazire, S., et al. Personality impressions based on Facebook profiles. *ICWSM 7* (2007), 1–4. pages 34

[77] Green, T. R. G., and Petre, M. Usability analysis of visual programming environments: A "cognitive dimensions" framework. *Journal of Visual Languages & Computing 7*, 2 (1996), 131–174. pages 139

[78] Green-Armytage, P. A colour alphabet and the limits of colour coding. *Journal of the International Colour Association 5* (2010). pages 57

[79] Grivet, S., Auber, D., Domenger, J.-P., and Melancon, G. Bubble tree drawing algorithm. In *Computer Vision and Graphics*. Springer, 2006, pp. 633–641. pages 57

[80] Groh, G. *Contextual Social Networking*. University of Technology, Munich, 2012. Habilitation thesis in Computer Science. pages 33

[81] Gross, J. J., Carstensen, L. L., Pasupathi, M., Tsai, J., Götestam Skorpen, C., and Hsu, A. Y. Emotion and aging: experience, expression, and control. *Psychology and Aging 12*, 4 (1997), 590. pages 106

[82] Gross, J. J., Richards, J. M., and John, O. P. Emotion regulation in everyday life. *Emotion regulation in families: Pathways to dysfunction and health* (2006), 13–35. pages 89, 105

[83] Gross, R., Acquisti, A., and Heinz, J. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society* (Alexandria, VA, USA, 2005), ACM, pp. 80, 71. pages 14

[84] Gürses, F. S. *Multilateral Privacy Requirements Analysis in Online Social Network Services*. PhD thesis, Informatics Section, Department of Computer Science, Faculty of Engineering Science, May 2010. pages 4, 5, 6

[85] Gürses, S., and Berendt, B. The social web and privacy: Practice, reciprocity and conflicts in social networks. *Privacy-Aware Knowledge Discovery* (2010), 395–432. pages 5, 16

[86] Hajian, S., Domingo-Ferrer, J., and Martinez-Balleste, A. Discrimination prevention in data mining for intrusion and crime detection. In *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on* (2011), IEEE, pp. 47–54. pages 126

[87] Hansen, M. Linkage control-integrating the essence of privacy protection into identity management. In *Proceedings of eChallenges* (2008), pp. 1585–1592. pages 15

[88] Harger, J. R., and Crossno, P. J. Comparison of open-source visual analytics toolkits. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (jan 2012), vol. 8294, International Society for Optics and Photonics. pages 140

[89] Harrower, M., and Brewer, C. A. Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal 40*, 1 (2003), 27–37. pages 192

[90] Healey, C. G., and Enns, J. T. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 145–167. pages 57

[91] Heer, J., Bostock, M., and Ogievetsky, V. A tour through the visualization zoo. *Communications of ACM 53*, 6 (2010), 59–67. pages 149

[92] Heer, J., and Shneiderman, B. Interactive dynamics for visual analysis. *Queue 10*, 2 (2012), 30. pages 30, 84, 150, 158, 159

[93] Hevia, A., and Micciancio, D. An indistinguishability-based characterization of anonymous channels. In *Privacy Enhancing Technologies* (2008), Springer, pp. 24–43. pages 17

[94] Hilderman, R. J., Hamilton, H. J., and Barber, B. Ranking the interestingness of summaries from data mining systems. In *FLAIRS Conference* (1999), Citeseer, pp. 100–106. pages 130

[95] Hogan, B. The presentation of self in the age of social media: Distinguishing performances and exhibitions online. *Bulletin of Science, Technology and Society 30*, 6 (Dec. 2010), 377–386. pages 34

[96] Huang, J., White, R., and Buscher, G. User see, user point: gaze and cursor alignment in web search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), ACM, pp. 1341–1350. pages 92

[97] Hutchins, E. Distributed cognition. *International Encyclopedia of the Social and Behavioral Sciences* (2000). pages 8

[98] Jameson, A., Berendt, B., Gabrielli, S., Cena, F., Gena, C., Vernero, F., and Reinecke, K. Choice architecture for human-computer interaction. *Foundations and Trends® in Human–Computer Interaction 7*, 1–2 (2013), 1–235. pages 4

[99] Johnson, B., and Shneiderman, B. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on* (1991), IEEE, pp. 284–291. pages 57, 135

[100] Kamvar, S. D., and Harris, J. We feel fine and searching the emotional web. In *Proceedings of the fourth ACM international conference on Web search and data mining* (2011), ACM, pp. 117–126. pages 28

[101] Kilmer, R., and Kilmer, W. O. *Designing interiors*. John Wiley & Sons, 2014. pages 59

[102] Klösgen, W. Explora: A multipattern and multistrategy discovery assistant. In *Advances in knowledge discovery and data mining* (1996), American Association for Artificial Intelligence, pp. 249–271. pages 107

[103] Knuth, D. E. Literate programming. *CSLI Lecture Notes, Stanford, CA: Center for the Study of Language and Information (CSLI), 1992 1* (1992). pages 169

[104] Kramer, A. D., Guillory, J. E., and Hancock, J. T. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences 111*, 24 (2014), 8788–8790. pages 105

[105] Kruskal, W. H., and Wallis, W. A. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association 47*, 260 (1952), 583–621. pages 114

[106] Kucuktunc, O., Cambazoglu, B. B., Weber, I., and Ferhatosmanoglu, H. A large-scale sentiment analysis for Yahoo! answers. In *Proceedings of the fifth ACM international conference on Web search and data mining* (2012), ACM, pp. 633–642. pages 106

[107] Kuhail, M. A., Lauesen, S., Pantazos, K., and Shangjin, X. Usability analysis of custom visualization tools. In *The Swedish Chapter of Eurographics (SIGRAD)* (2012), A. Kerren and S. Seipel, Eds., vol. 81 of *Linköping Electronic Conference Proceedings*, Linköping University Electronic Press, pp. 19–28. pages 140

[108] Kuntz, P., Guillet, F., Lehn, R., and Briand, H. A user-driven process for mining association rules. In *Principles of data mining and knowledge discovery*. Springer, 2000, pp. 483–489. pages 128

[109] Lammarsch, T., Aigner, W., Bertone, A., Miksch, S., Turic, T., and Gartner, J. A comparison of programming platforms for interactive

visualization in web browser based applications. In *The 12th International Conference on Information Visualisation* (2008), IEEE, pp. 194–199. pages 140

[110] Lampe, C., Ellison, N., and Steinfield, C. A face (book) in the crowd: Social searching vs. social browsing. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (2006), ACM, pp. 167–170. pages 34

[111] Laramee, R. S. Comparing and evaluating computer graphics and visualization software. *Software: Practice and Experience 38*, 7 (2008), 735–760. pages 140

[112] Lederer, S., Hong, J. I., Dey, A. K., and Landay, J. A. Personal privacy through understanding and action: five pitfalls for designers. *Personal and Ubiquitous Computing 8*, 6 (2004), 440–454. pages 15

[113] Leman, D., Feelders, A., and Knobbe, A. Exceptional model mining. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 1–16. pages 117

[114] Lewis, J. R. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction 7*, 1 (1995), 57–78. pages 90

[115] Lipford, H. R., Besmer, A., and Watson, J. Understanding privacy settings in facebook with an audience view. *UPSEC 8* (2008), 1–8. pages 101

[116] Liu, B., Hsu, W., Mun, L.-F., and Lee, H.-Y. Finding interesting patterns using user expectations. *Knowledge and Data Engineering, IEEE Transactions on 11*, 6 (1999), 817–832. pages 130

[117] Liu, H., Maes, P., and Davenport, G. Unraveling the taste fabric of social networks. *International Journal on Semantic Web and Information Systems 2*, 1 (2008). pages 15

[118] Lui, M., and Baldwin, T. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations* (2012), Association for Computational Linguistics, pp. 25–30. pages 109

[119] Macrae, C. N., and Bodenhausen, G. V. Social cognition: Thinking categorically about others. *Annual review of psychology 51*, 1 (2000), 93–120. pages 34, 35

[120] Mann, H. B., and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60. pages 114

[121] Marco Lui, T. B. Accurate language identification of Twitter messages. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)@ EACL* (2014), pp. 17–25. pages 109

[122] Marwick, A. E., et al. I tweet honestly, I tweet passionately: Twitter users, context collapse, and the imagined audience. *New media & society 13*, 1 (2011), 114–133. pages 33, 35

[123] Mazzia, A., LeFevre, K., and Adar, E. The PViz comprehension tool for social network privacy settings. In *Proceedings of the Eighth Symposium on Usable Privacy and Security* (2012), ACM, p. 13. pages 18, 45, 101

[124] McAuley, J. J., and Leskovec, J. Learning to discover social circles in ego networks. In *NIPS* (2012), pp. 548–556. pages 73, 74, 75, 76

[125] Miller, G. A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review 63*, 2 (1956), 81. pages 54

[126] Momtazi, S. Fine-grained German Sentiment Analysis on Social Media. In *Proceedings of the 8th International Conference on Language Resources and Evaluation* (2012), pp. 1215–1220. pages 113

[127] Moore, K., and McElroy, J. C. The influence of personality on Facebook usage, wall postings, and regret. *Computers in Human Behavior 28*, 1 (2012), 267–274. pages 65

[128] Moreira, A. A., Paula, D. R., Costa Filho, R. N., and Andrade Jr, J. S. Competitive cluster growth in complex networks. *Physical Review E 73*, 6 (2006), 065101. pages 80

[129] Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics 5*, 1 (1957), 32–38. pages 70, 76

[130] Munzner, T. *Visualization Analysis and Design*. CRC Press, 2014. pages 141, 143, 146, 149

[131] Nakatani, S. Language Detection Library for Java. `https://code.google.com/p/language-detection/`, 2011. pages 109

[132] Newman, M. E. A measure of betweenness centrality based on random walks. *Social networks 27*, 1 (2005), 39–54. pages 87

[133] Newman, M. E. Finding community structure in networks using the eigenvectors of matrices. *Physical review E 74*, 3 (2006), 036104. pages 74

[134] Newman, M. E. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences 103*, 23 (2006), 8577–8582. pages 43, 44, 62, 74, 200

[135] Nguyen, D. H., and Mynatt, E. D. Privacy mirrors: understanding and shaping socio-technical ubiquitous computing systems. *Atlanta, Technical Report* (2002). pages 15

[136] Nguyen, Q. V., and Huang, M. L. A space-optimized tree visualization. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on* (2002), IEEE, pp. 85–92. pages 57

[137] Patil, S., and Kobsa, A. Enhancing privacy management support in instant messaging. *Interacting with Computers 22*, 3 (2010), 206–217. pages 23

[138] Pedreshi, D., Ruggieri, S., and Turini, F. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 560–568. pages xviii, 125, 127, 128

[139] Phillips, D. J. Privacy policy and PETs The influence of policy regimes on the development and social implications of privacy enhancing technologies. *New Media & Society 6*, 6 (2004), 691–706. pages 5

[140] Raynes-Goldie, K. Aliases, creeping, and wall cleaning: Understanding privacy in the age of Facebook. *First Monday 15*, 1 (2010). pages 4, 33, 35

[141] Rodrigues, E. M., Milic-Frayling, N., Smith, M., Shneiderman, B., and Hansen, D. Group-in-a-box layout for multi-faceted analysis of communities. In *Proceedings of the 3rd IEEE International Conference on Social Computing* (2011), pp. 354–361. pages 36

[142] Ruggieri, S., Pedreschi, D., and Turini, F. Data mining for discrimination discovery. *ACM Transactions on Knowledge Discovery from Data (TKDD) 4*, 2 (2010), 9. pages 125, 129

[143] Ruggieri, S., Pedreschi, D., and Turini, F. Dcube: Discrimination discovery in databases. In *Proceedings of the 2010 ACM SIGMOD*

International Conference on Management of data (2010), ACM, pp. 1127–1130. pages 126, 201

[144] Satyanarayan, A., Wongsuphasawat, K., and Heer, J. Declarative interaction design for data visualization. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (2014), ACM, pp. 669–678. pages 153

[145] Sayaf, R., and Clarke, D. Access control models for online social networks. *Social Network Engineering for Secure Web Data and Services* (2012), 32–65. pages 14

[146] Sayaf, R., Clarke, D., and Harper, R. CPS2: a Contextual Privacy Framework for Social Software. In *10th International Conference on Security and Privacy in Communication Networks (SECURECOMM2014)* (2014). pages 4, 14

[147] Schilit, B., Adams, N., and Want, R. Context-aware computing applications. In *Proceedings of the 1st Workshop on Mobile Computing Systems and Applications* (Washington, DC, USA, 1994), WMCSA '94, IEEE Computer Society, pp. 85–90. pages 33

[148] Seo, J., and Shneiderman, B. Interactively exploring hierarchical clustering results [gene identification]. *Computer 35*, 7 (2002), 80–86. pages 56

[149] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research 13*, 11 (2003), 2498–2504. pages 36

[150] Shapiro, J. J., and Hughes, S. K. Information literacy as a liberal art? *Educom review 31* (1996), 31–35. pages 2

[151] Shapiro, S. S., and Wilk, M. B. An analysis of variance test for normality (complete samples). *Biometrika* (1965), 591–611. pages 108

[152] Shearer, C. The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing 5*, 4 (2000), 13–22. pages 6

[153] Shneiderman, B. *Designing the user interface: strategies for effective human-computer interaction*, vol. 3. Addison-Wesley Reading, MA, 1992. pages 180

[154] Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages* (1996), IEEE, pp. 336–343. pages 30, 158, 179

[155] Shneiderman, B. Inventing discovery tools: Combining information visualization with data mining. In *Discovery Science* (2001), Springer, pp. 17–28. pages 9

[156] Shneiderman, B., and Dunne, C. Interactive network exploration to derive insights: filtering, clustering, grouping, and simplification. In *Graph Drawing* (2013), Springer, pp. 2–18. pages 36, 84

[157] Shneiderman, B., and Wattenberg, M. Ordered treemap layouts. In *Proceedings of the IEEE Symposium on Information Visualization 2001* (2001), vol. 73078. pages 57

[158] Siganos, A., Vagenas-Nanos, E., and Verwijmeren, P. Facebook's daily sentiment and international stock markets. *Journal of Economic Behavior & Organization 107* (2014), 730–743. pages 105

[159] Silva, S., Santos, B. S., and Madeira, J. Using color in visualization: A survey. *Computers & Graphics 35*, 2 (2011), 320–333. pages 192

[160] Simon, R. W., and Nath, L. E. Gender and emotion in the United States: Do men and women differ in self-reports of feelings and expressive behavior? 1. *American journal of sociology 109*, 5 (2004), 1137–1176. pages 106

[161] Stevens, S. S. On the psychophysical law. *Psychological review 64*, 3 (1957), 153. pages 58

[162] Stone, A. A., Schwartz, J. E., Broderick, J. E., and Deaton, A. A snapshot of the age distribution of psychological well-being in the United States. *Proceedings of the National Academy of Sciences 107*, 22 (2010), 9985–9990. pages 106, 111

[163] Sutcliffe, A., Dunbar, R., Binder, J., and Arrow, H. Relationships and the social brain: Integrating psychological and evolutionary perspectives. *British journal of psychology 103*, 2 (2012), 149–168. pages 89

[164] Sy, B. K. Discovering association patterns based on mutual information. In *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2003, pp. 369–378. pages 130

[165] Taylor, J. *Emotional experience and romantic relationship status in emerging adult college women and men*. Colorado State University, 2009. pages 107

[166] Thelwall, M., Buckley, K., and Paltoglou, G. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology 63*, 1 (2012), 163–173. pages 112

[167] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology 61*, 12 (2010), 2544–2558. pages 112

[168] Thelwall, M., Wilkinson, D., and Uppal, S. Data mining emotion in social network communication: Gender differences in MySpace. *Journal of the American Society for Information Science and Technology 61*, 1 (2010), 190–199. pages 106

[169] Tong, S. T., Van Der Heide, B., Langwell, L., and Walther, J. B. Too much of a good thing? The relationship between number of friends and interpersonal impressions on Facebook. *Journal of Computer-Mediated Communication 13*, 3 (2008), 531–549. pages 34

[170] Tukey, J. W. Box-and-whisker plots. *Exploratory Data Analysis* (1977), 39–43. pages 107

[171] Turini, D. P. S. R. F. Measuring discrimination in socially-sensitive decision records. In *SDM* (2009), SIAM, pp. 581–592. pages 126

[172] Ugander, J., Karrer, B., Backstrom, L., and Marlow, C. The anatomy of the Facebook social graph. *arXiv preprint arXiv:1111.4503* (2011). pages 80

[173] van Leeuwen, M., and Knobbe, A. Diverse subgroup set discovery. *Data Mining and Knowledge Discovery 25*, 2 (2012), 208–242. pages 117

[174] Van Wijk, J. J., and Van de Wetering, H. Cushion treemaps: Visualization of hierarchical information. In *Proceedings of IEEE Symposium on Information Visualization* (1999), IEEE, pp. 73–78. pages 57

[175] Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. State-of-the-art in privacy preserving data mining. *ACM Sigmod Record 33*, 1 (2004), 50–57. pages 4

[176] Vural, A. G., Cambazoglu, B. B., Senkul, P., and Tokgoz, Z. O. A framework for sentiment analysis in Turkish: Application to polarity detection of movie reviews in Turkish. In *Computer and Information Sciences III*. Springer, 2013, pp. 437–445. pages 113

[177] Wang, L., Giesen, J., McDonnell, K. T., Zolliker, P., and Mueller, K. Color design for illustrative visualization. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (2008), 1739–1754. pages 58

[178] Wang, Y., Norcie, G., Komanduri, S., Acquisti, A., Leon, P. G., and Cranor, L. F. I regretted the minute I pressed share: A qualitative study of regrets on Facebook. In *Proceedings of the Seventh Symposium on Usable Privacy and Security* (2011), ACM, p. 10. pages 65, 101

[179] Wang Baldonado, M. Q., Woodruff, A., and Kuchinsky, A. Guidelines for using multiple views in information visualization. In *Proceedings of the working conference on advanced visual interfaces* (2000), ACM, pp. 110–119. pages 180

[180] Ware, C. *Information visualization: perception for design*. Elsevier, 2012. pages 8, 147

[181] Ware, C., Purchase, H., Colpoys, L., and McGill, M. Cognitive measurements of graph aesthetics. *Information Visualization 1*, 2 (2002), 103–110. pages 59

[182] Warren, S. D., and Brandeis, L. D. The right to privacy. *Harvard law review* (1890), 193–220. pages 5

[183] Wilcoxon, F., Katti, S., and Wilcox, R. A. *Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test*. American Cyanamid Comp., 1963. pages 91

[184] Wolf, R. D., Heyman, R., and Pierson, J. Privacy by Design through social requirements analysis of social network sites from a user perspective. In *European Data Protection: Coming of Age*, S. Gutwirth, R. Leenes, and P. de Hert, Eds. Springer, Berlin, Germany, 2012. pages 34

[185] Wong, P. C., Whitney, P., and Thomas, J. Visualizing association rules for text mining. In *Proceedings of IEEE Symposium on Information Visualization* (1999), IEEE, pp. 120–123. pages 128

[186] Wrobel, S. An algorithm for multi-relational discovery of subgroups. In *Principles of Data Mining and Knowledge Discovery*. Springer, 1997, pp. 78–87. pages 107

[187] Yahia, S. B., and Nguifo, E. M. Emulating a cooperative behavior in a generic association rule visualization tool. In *The 16th IEEE International Conference on Tools with Artificial Intelligence* (2004), IEEE, pp. 148–155. pages 131

[188] Yap, S. C., Anusic, I., and Lucas, R. E. Does personality moderate reaction and adaptation to major life events? Evidence from the british household panel survey. *Journal of research in personality 46*, 5 (2012), 477–488. pages 106

[189] Yi, J. S., ah Kang, Y., Stasko, J. T., and Jacko, J. A. Toward a deeper understanding of the role of interaction in information visualization. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6 (2007), 1224–1231. pages 84

[190] Zammuner, V. L. Men's and women's lay theories of emotion. *Gender and emotion: Social psychological perspectives* (2000), 48. pages 106

[191] Zhao, S., Grasmuck, S., and Martin, J. Identity construction on Facebook: Digital empowerment in anchored relationships. *Computers in human behavior 24*, 5 (2008), 1816–1836. pages 34

[192] Zuckerberg, M. One Billion People on Facebook, 2012. Retrieved Jul 19, 2014. pages 32

Curriculum Vitae

Bo Gao was born in Ulanhot, Nei Mongol, China on January 11th 1987. He started his higher education in Beijing Jiaotong University (BJTU) in 2005, majored in software engineering. One year later, he went to study as an exchange student in Group T university college in Leuven. In 2009, he obtained a Bachelor degree in software engineering from BJTU and a Master degree in electronic engineering, specialized in internet computing from Group T. He then went on to study Artificial Intelligence in KU Leuven, and obtained the corresponding advanced Master degree in 2010. In the spring of 2011, he started his doctoral studies in the department of computer science of KU Leuven, under the supervision of Prof.Bettina Berendt. In December 2015, he will defend his thesis, titled "Exploratory Visualization Design towards Online Social Network Privacy and Data Literacy".

List of publications

Journal Articles

De Wolf, Ralf; Gao, Bo; Berendt, Bettina; Pierson, Jo. The promise of audience transparency: Exploring users' perceptions and behaviors towards visualizations of networked audiences on Facebook, Telematics and Informatics, volume 32, issue 4, pages 890-908, 2015.

Van Rijn, Jan; Bischl, Bernd; Torgo, Luis; Gao, Bo; Umaashankar, Venkatesh; Fischer, Simon; Winter, Patrick; Wiswedel, Bernd; Berthold, Michael; Vanschoren, Joaquin. OpenML: A collaborative science platform, Lecture Notes in Computer Science, volume 8190, pages 645-649, 2013.

Book Chapter

Gao, Bo, and Bettina Berendt. "Friends and Circles – A Design Study for Contact Management in Egocentric Online Social Networks." In Online Social Media Analysis and Visualization, pp. 129-161. Springer International Publishing, 2014.

Conferences and Workshops

Gao, Bo; Berendt, Bettina; Vanschoren, Joaquin. Who is more positive in private? Analyzing sentiment differences across privacy levels and demographic factors in Facebook chats and posts, ASONAM, Paris, France, 25-28 August, 2015, Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pages 605-610.

Gao, Bo; Berendt, Bettina. Circles, posts and privacy in egocentric social

networks: An exploratory visualization approach, ASONAM, Niagara Falls, Canada, 25-28 August 2013, Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pages 792-796.

Gao, Bo; Berendt, Bettina; Clarke, Dave; De Wolf, Ralf; Peetz, Thomas; Pierson, Jo; Sayaf, Rula. Interactive grouping of friends in OSN: Towards online context management, ICDM Workshops, Brussels, Belgium, 10 December 2012, 2012 IEEE 12th International Conference on Data Mining Workshops, pages 555-562.

Gao, Bo; Berendt, Bettina. Visual data mining for higher-level patterns: Discrimination-aware data mining and beyond, Benelearn, The Hague, May 20 2011, Benelearn 2011. Proceedings of the Twentieth Belgian Dutch Conference on Machine Learning, pages 45-52.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
MACHINE LEARNING RESEARCH GROUP
Celestijnenlaan 200A box 2402
B-3001 Leuven
bo.gao@cs.kuleuven.be
http://people.cs.kuleuven.be/~bo.gao